CD CONTROL DATA
CORPORATION

CYBER CROSS SYSTEM
VERSION 1
MICRO ASSEMBLER
REFERENCE MANUAL

CONTROL DATA®
CYBER 170 SERIES
CYBER 70 SERIES MODELS 72, 73, 74
6000 SERIES COMPUTER SYSTEMS
CYBER 18 COMPUTER SYSTEMS
255X HOST COMMUNICATIONS PROCESSORS

REVISION RECORD

REVISION	DESCRIPTION
A	Manual released.
(12/75)	
B	Manual revised to reflect NOS/BE 1.1 changes and manual title change.
(4/76)	
C	Manual revised to include PSR 45754, adding pseudo instructions LST and NLS. Pages revised: ii, iii, vii, 3-3,
(5/77)	3-5, 3-6, 5-4, A-1, B-1, B-2, B-3, E-3, comment sheet. Page added: 5-4.1/5-4.2.

Publication No.
96836400

© 1975, 1976, 1977
by Control Data Corporation

Printed in the United States of America

Address comments concerning this
manual to:
Control Data Corporation
Publications and Graphics Division
4455 Eastgate Mall
La Jolla, California 92037
or use Comment Sheet in the back of
this manual.

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual, are indicated by bars in the margins or by a dot near the page number If the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV	PAGE	REV
Cover	--								
Title Page	--								
ii	C								
iii	C								
v	B								
vii	C								
viii thru x	B								
1-1	B								
2-1	B								
2-2	B								
3-1	B								
3-2	B								
3-3	C								
3-4	B								
3-5	C								
3-6	C								
3-7	B								
4-1	B								
5-1 thru 5-3	B								
5-4	C								
5-4.1/5-4.2	C								
5-5 thru 5-12	B								
6-1 thru 6-11	B								
7-1 thru 7-9	B								
8-1	B								
8-2	B								
9-1 thru 9-4	B								
10-1	B								
10-2	B								
11-1 thru 11-4	B								
Glossary-1	B								
A-1	C								
B-1 thru B-3	C								
C-1 thru C-3	B								
D-1 thru D-3	B								
E-1	B								
E-2	B								
E-3	C								
E-4	B								
E-5	B								
F-1	B								
G-1	B								
G-2	B								
H-1	B								
Index-1 thru Index-3	B								
Comment Sheet	C								
Cover	--								

PREFACE

The CYBER Micro Assembler is a component of the CONTROL DATA® CYBER Cross System. The Micro Assembler operates under control of the CYBER 170/70/6000 NOS or NOS/BE operating system. It is intended to assemble micro code for the CYBER 18 computer series and the CDC 255x Series Host Communications Processors. A separate version of the Micro Assembler is available for the CYBER 18 computer series.

This manual describes the general operation of the assembler and provides the necessary instructions for preparing programs for assembly. No attempt is made here to provide a programmers guide and, therefore, examples are limited. It is assumed that the reader is already familiar with the operation of the CYBER 18 computer.

Information applicable to the Host Operating System can be found in the Literature Distribution Services catalog. Additional information can be found in the following publications:

Description	Publication No.
CYBER Cross System Version 1 Reference Manual	96836000
CYBER Cross System Version 1 Macro Assembler Reference Manual	96836500
CYBER Cross System Version 1 Link Editor and Library Maintenance Programs Reference Manual	60471200
NOS/BE 1 Reference Manual	60493800
NOS 1 Reference Manual, Volume 1	60435400
NOS 1 Reference Manual, Volume 2	60445300
Micro-Programmable Computer Family Micro Processor Reference Manual	88973400
1700 Enhanced Micro Processor with Core Memory Reference Manual	88973500

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

FIGURES

TABLES

INTRODUCTION 1

The assembler for the CYBER 18 computer series and CDC 255x processors provides the mnemonic language necessary for the programmer to write a micro program. The assembler translates symbolic source program instructions into object machine instructions and provides a listing of assembly results.

The characteristics of the assembler as written for the CYBER 170/70/6000 and CYBER 18 Series computers are described. This assembler is based on the MICRO-71 assembler for the MPP computer.

Input to this assembler consists of one or more source programs followed by a FINIS card. Each program begins with an IDENT card and is terminated with an END card. Each program is coded using these basic elements:

    Symbols
    Constants
    Pseudo instructions
    Mnemonic instructions

The basic elements are punched into a card in specific fields, always left-justified within the field.

Output from the assembler consists of the following:

- Assembly listing including diagnostics
- Zero location map
- Origin location map
- Relocatable object image
- Deadstart object image

A source input statement to the assembler consists of eleven fields as shown in table 2-1 and as illustrated in the coding form shown in figure 2-1. Of these fields, the Q (qualifier), location, and comment fields are used to improve the documentation of the assembled micro instructions. The eight fields used on the input form are in the same order that the programmer will tend to use in preparing micro instructions for a micro program.

Information entered in each field (if anything is entered) is entered left-justified with a blank fill. Information that is not entered left-justified is not processed correctly by the assembler.

TABLE 2-1. SOURCE STATEMENT FIELDS

FIELDS	COLUMNS	COMMENTS
Q (qualifier)	1	The qualifier field may specify whether the statement is a comment, an upper instruction, or a lower instruction.
Location	2 through 9	The location field specifies the statement's symbolic address in this program.
F (function)	11 through 16	The function field specifies a logical, arithmetic, shift, or scale operation that is performed by the arithmetic and logic unit (ALU) on two sources and placed in a destination.
A	17 through 22	Specifies the A source of the function
B	23 through 28	Specifies the B source of the function
D	29 through 34	Specifies the destination of the result of the ALU
S (special)	35 through 40	The special field provides special instruction modes that either: • Extend the A, B, and D fields, or • Provide a special command which is performed in parallel with the data transfers taking place in the ALU.
C (constant)	41 through 49	The constant field specifies another special command that is performed independently of the rest of the instruction; it is executed in parallel with the rest of the instruction.
M (mode)	50	The mode field specifies the addressing method for obtaining the next instruction pair: sequential, jump, or return.
T (test)	51 through 55	The test field is the conditional branch of the instruction and specifies which instruction (upper or lower) of the next instruction pair to execute. The test and branch are executed after the rest of the instruction has executed.
Comment	56 through 80	The comment field is used for remarks that are printed as part of the list output.

CONTROL DATA CORPORATION

PROGRAM ROUTING

MICRO PROCESSOR CODING FORM

NAME _____

PAGE _____

DATE _____

Figure 2-1. Micro Processor Coding Form

BASIC ELEMENTS 3

The basic elements processed by the assembler are symbols, constants, pseudo instructions, and mnemonic instructions.

3.1 SYMBOLS

A symbol is a one- to eight-character name that may be used as:

- A location label

- An alternate representation for a constant

A symbol is defined when it appears in the location field of the input form (columns 2 through 9). When a symbol appears in the location field, it is used to name the location of a portion of the program or data storage, or it is used in an EQU pseudo instruction to define the symbol as equivalent to the item defined to the right of the EQU function code.

A symbol may be used in the location field (columns 2 through 9), the S field (columns 35 through 40), or in the C field (columns 41 through 49) of the input form.

A symbol is undefined when it has never appeared in the location field of the input form, or if it is equated to an undefined symbol. The assembler identifies the use of undefined symbols on the assembly listing.

A symbol consists of any combination of one to eight 026 keypunch characters (the 48-character set) except the slash (/), the equate sign (=), the plus (+), the minus (-), or the asterisk (*). Several examples of legal and illegal symbols are shown below. A symbol must contain a non-numeric character to separate it from a constant.

Examples:

HCNYL	Legal
TAG	Legal
1234	Illegal (will be interpreted as a constant)
*12.3	Illegal (contains an asterisk)
XYZ/3P	Illegal (contains a slash)
B = 3	Illegal (contains an equal sign)

3.2 CONSTANTS

Constants are used to represent numbers and may be used in the S and C fields of the input form. Constants may also be used on the right side of the EQU pseudo instruction. The assembler recognizes three types of numeric constants: decimal, octal, and hexadecimal. The numeric constant is represented by a string of digits within the number base of the constant. Decimal constants have no suffix, octal constants have B as a suffix, and hexadecimal constants have X as a suffix. Constants must be in the range of C through 4095.

3.2.1 DECIMAL CONSTANTS

A decimal constant consists of a string of decimal digits. If the constant is larger than the field width of the micro instruction, the high order bits will be discarded.

Examples:

999	Legal
98A	Illegal (contains an alphabetic character)
12.1	Illegal (contains a decimal point)

3.2.2 OCTAL CONSTANTS

An octal constant consists of a string of octal digits that are suffixed with the letter B.

Examples:

123B	Legal
77B	Legal
019B	Illegal (contains a non-octal digit)

3.2.3 HEXADECIMAL CONSTANTS

A hexadecimal constant consists of a string of hexadecimal digits and is suffixed with the letter X. The hexadecimal digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

Examples:

77BX	Legal
1GX	Illegal (contains a non-hexadecimal digit)
ABC	Illegal (has no X suffix)
77B	Will be interpreted as an octal 77

3.3 PSEUDO INSTRUCTIONS

Pseudo instructions direct the assembler to perform specific functions. They do not generate micro processor instructions. They define assembler control, listing control, data definition, and other operations. Pseudo instructions are defined in section 5 of this manual.

3.4 MNEMONIC INSTRUCTIONS

Mnemonic instructions allow the programmer to use convenient names to specify the binary information to be inserted in each field of the micro instruction. This list of mnemonic instructions recognized by the assembler for each field of the instruction is given in table 3-1. Detailed usage of the mnemonic instructions is given in section 6 of this manual.

TABLE 3-1. MNEMONIC MACHINE INSTRUCTIONS

T MNEMONIC	MACHINE CODE	BIT 24	T MNEMONIC	MACHINE CODE	BIT 24
*L	0	0	LQL	1	1
U	1	0	K7L	2	1
L	2	0	OVFL	3	1
KZU	3	0	BTU	4	1
NZU	4	0	LQ*L	5	1
INTU	5	0	BTU*	5	1
NU	6	0	COL	6	1
ZL	7	0	Z*L	7	1

TABLE 3-1. MNEMONIC MACHINE INSTRUCTIONS (Continued)

F CODE ARITHMETIC & LOGICAL MNEMONIC	HEXADECIMAL MACHINE CODE	SHIFT AND SCALE MNEMONIC	HEXADECIMAL MACHINE CODE	IMPLIED A CODE	IMPLIED B CODE
-A†	0	ALOE	1E	2	0
-A+-B††	1	AL0E	1E	2	0
-A+B	2	AQLOE	1E	3	0
ONE	3	AQL0E	1E	3	0
-A.-B†††	4	AROE	1E	4	0
-B	5	AR0E	1E	4	0
-EOR	6	AQROE	1E	5	0
A+-B	7	AQR0E	1E	5	0
-A.B	8	AL1E	1E	2	1
EOR	9	ARSE	1E	4	1
B	A	AQRSE	1E	5	1
A+B	B	ALEA	1E	2	2
ZERO	C	AQLEA	1E	3	2
A.-B	D	AREA	1E	4	2
A.B	E	AQREA	1E	5	2
A	F	SLOE	1F	2	0
SUB	14	SL0E	1F	2	0
SUBT	15	SDLOE	1F	3	0
SUB-	16	SDL0E	1F	3	0
SUB-C	16	SL1E	1F	2	1
SUB-T	17	SLEA	1F	2	2
SUB-TC	17	SDLEA	1F	3	2
ADD	18				
ADDT	19				
ADD+	1A				
ADD+T	1B				

†Minus (-) means bit-by-bit complement
††Plus (+) means inclusive OR
†††Period (.) means AND

TABLE 3-1. MNEMONIC MACHINE INSTRUCTIONS (Continued)

A MNEMONIC	MACHINE CODE	IMPLIED S	B MNEMONIC	MACHINE CODE	IMPLIED S	HEXADECIMAL IMPLIED C
F2	0		F2	0		
P	1		ZERO	1		C
I	2		N	1		8
X	3		K	1		4
A	4		N, K	1		0
F	5		BG	2		
F1	6		X	3		
XF	6		Q	4		
MEM	7		F	5		
SM1	0	7	F1	6		
M1	1	7	XF	6		
SM2	2	7	MEM	7		
R2	2	7	CRTJ	1	8	
M2	3	7	INRD	2	8	
R3	3	7	INRS	3	8	
A*R8	4	7	MMU	4	8	
R5	5	7	MML	5	8	
RA	5	7	INTA	6	8	
FN1	5	7	FMT0	0	8	
X*	6	7	FMT1	1	8	
GR	6	7	FMT2	2	8	
R6	6	7	FMT3	3	8	
Q*	7	7				
R7	7	7				
RQ	7	7				
FN2	7	7				

TABLE 3-1. MNEMONIC MACHINE INSTRUCTIONS (Continued)

D MNEMONIC	MACHINE CODE	HEXADECIMAL IMPLIED S	D MNEMONIC	MACHINE CODE	HEXADECIMAL IMPLIED S
NOP	0		T5	5	B
P	1		X*	6	B
I	2		T6	6	B
Q	3		GR	6	B
F1	4		Q*	7	B
A	5		T7	7	B
X	6				
F	7				

D MNEMONIC	MACHINE CODE	HEXADECIMAL IMPLIED S	S MNEMONIC	HEXADECIMAL MACHINE CODE
AA*	5	1	NOP	0
XX*	6	1	DD	1
XGR	6	1	RPT	2
FQ*	7	1	READ	3
IOD	0	9	WRITE	4
RA	0	9	L8EA	5
FN1	0	9	F2WR	6
IOA	1	9	AP	7
RQ	1	9	BP	8
FN2	1	9	DP	9
MMU	2	9	APDP	A
MML	3	9	DPP	B
M1	4	9	GATEI	C
SM1	5	9	HALT	D
M2	6	9	RTJ	E
T3	6	9	CLRNP	F
SM2	7	9		
T2	7	9		
A*LH	1	B		
X*LH	2	B		
Q*LH	3	B		
T4	4	B		
A*	5	B		

TABLE 3-1. MNEMONIC MACHINE INSTRUCTIONS (Continued)

C MNEMONIC	HEXADECIMAL MACHINE CODE	BIT 19	WIDTH OF VARIABLE FOR C	C MNEMONIC	HEXADECIMAL MACHINE CODE	BIT 19	WIDTH OF VARIABLE FOR C
K=	0	0	8	RL0E	74	0	
WRCH/	20	0	2	RLOE	74	0	
RMW	24	0		RL1E	75	0	
WRHW0	25	0		RR0E	76	0	
WRHW1	27	0		RROE	76	0	
WRPB	28	0		RR1E	77	0	
GATEIXT	30	0		TMA/	0	1	4
CLRK	40	0		TMAK/	10	1	4
DECK	44	0		GITMAK/	20	1	3
INCK	45	0		GITMAK/XT	2C	1	
CLRN	48	0		TK/	30	1	4
DECK	4C	0		TN/	40	1	4
INCN	4D	0		SUB	50	1	
SETF/	50	0	4	SLB	60	1	
CLRF/	60	0	4	N=	0	1	8
RQLXN	70	0					
RQR0E	72	0					
RQROE	72	0					
RQR1E	73	0					

A slash (/) or an equal sign (=) following the mnemonic implies a constant will follow the mnemonic and must have a value between 0 and 2^W-1 where W is the width of the variable for the C field; e.g., SETF/12.

LOCATION FIELD 4

The location field is coded in columns 1 through 9 of the input coding form and consists of two subfields:

- The Q (qualifier) field
- The location label field

4.1 Q FIELD

The Q field is column 1 of the input coding form. It is used to specify the nature of the rest of the statement and to provide a fine grain location of the resulting micro instruction within a micro-memory address. This field may contain an asterisk (*), dollar sign ($), plus sign (+), minus sign (-), or it may be blank.

An asterisk or dollar sign specifies that the rest of the input source statement is a remark and that the remaining 79 columns contain comments. This qualifier allows the remarks card to be printed on the listing with no effect on the assembler object code output.

A plus in the Q field locates the resulting micro instruction as the upper instruction of a micro-instruction pair.

A minus in the Q field locates the resulting micro instruction as the lower instruction of a micro-instruction pair.

A blank in the Q field locates the resulting micro instruction in the next available half of a micro-instruction pair.

4.2 LOCATION LABEL FIELD

The location label field is in columns 2 through 9 of the input coding form. This field may be left blank or it may contain a symbol. If a symbol is included in the field, it must be entered left-justified and follow the definition of a symbol.

The location label field with a symbol is used to assign a mnemonic address to the corresponding micro instruction, or it may be used in the EQU pseudo instruction to assign a value to the symbol in the label field.

A symbol in the label field of a micro instruction takes on the upper/lower quality of the actual micro instruction location. This quality is used in coding jumps in the C field of a micro instruction.

Pseudo instructions are instructions to the assembler and normally do not result in any micro-code output (the only exceptions are the HEX, DEC, and OCT pseudo instructions). A pseudo instruction consists of the pseudo operation code, which is coded in the F field of the input form (columns 11 through 16), plus additional information coded in the other fields of the input form. The detailed field usage is given under each pseudo instruction.

5.1 ASSEMBLER CONTROL PSEUDO INSTRUCTIONS

These pseudo instructions define and control the operation of the MP assembler, but do not generate code in the object program.

5.1.1 IDENT

This pseudo instruction provides program identification and must be used as the first instruction of each program. The text in columns 17 through 80 of the card with the IDENT operation code is listed as the first line at the top of each page of the output listing. In addition, if the RELO pseudo instruction is used in this same program, data from columns 17 through 22 will be used as the name in the NAM block, and data in columns 23 through 66 will be used in the NAM block as ID information.

Example:

	11	17
	IDENT	CDC 844 DISK FILE CONTROLLER EMULATION FOR MP

5.1.2 CPR

This pseudo instruction causes a copyright notice to be printed as the second line of each page of the listing. The four decimal digits contained in columns 17 through 20 of this pseudo instruction are included in the notice.

The following example provides a listing output of

COPYRIGHT 1976 CONTROL DATA CORPORATION

as the second line of each page of listing.

```
          |11        |17
_____|_____|_____
        / |          |
       /  |CPR       |1976
      (   |          |
```

5.1.3 END

The END pseudo instruction signals the end of this program for assembly and must be the last instruction in a program. It causes the assembler to proceed with the complete assembly process. On completion of the assembly process, the assembler is reset and continues reading input information to obtain the next micro program of a batch to assemble. The total assembly process is completed on detecting a FINIS pseudo instruction.

Example:

```
          |11
_____|_____
        / |
       /  |END
      (   |
          |
```

5.1.4 FINIS

The FINIS pseudo instruction signals the completion of a batch of assemblies by the assembler and returns control to the host computer operating system.

Example:

```
          |11
_____|_____
        / |
       /  |FINIS
      (   |
          |
```

5.2 LISTING CONTROL PSEUDO INSTRUCTIONS

The listing output for the assembler is controlled by the following pseudo instructions. These pseudo instructions may appear anywhere in the source input between IDENT and END pseudo instructions.

5.2.1 COMMENT CARD

Any source card with an asterisk (*) or a dollar sign ($) in column 1 is treated as a comment card. All columns of the comment card are printed.

5.2.2 EJECT

The EJECT pseudo instruction causes the listing to eject to the top of the next page, and the next instruction will be printed following the title line on the next page. The EJECT pseudo instruction card is not printed.

Example:
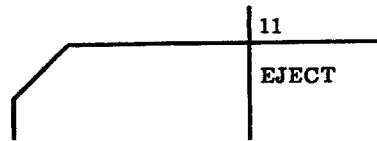
5.2.3 SPACE

The SPACE pseudo instruction causes blank lines to be printed. The SPACE pseudo instruction is not printed. The number of blank lines to be listed is defined in the A field of the pseudo instruction. The A field may contain a constant or a predefined symbol, as in the following example:
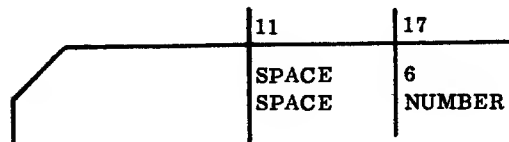
The first SPACE pseudo instruction would cause six blank lines to be printed. The second would cause two lines to be printed if NUMBER had been defined to be 2.

5.2.4 BOX

This pseudo instruction is used in conjunction with EBOX to provide emphasis for comments in the listing. This pseudo instruction is not printed; however, a card of asterisks will be listed. All succeeding cards will have asterisks in columns 1 and 80 to create comment cards. Only an EBOX pseudo instruction following a BOX pseudo instruction will be executed. The listing will be spaced one line before printing the first line of asterisks for the BOX command.

NOTE

A BOX command will turn all succeeding micro instructions to comment cards until EBOX is encountered.

Example:

```
          | 11
 /_____|_____
/         |
|         | BOX
|         |
(         |
```
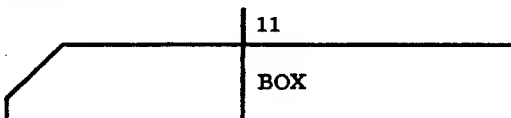
5.2.5 EBOX

This pseudo instruction causes a card of asterisks to be listed rather than this pseudo instruction. In addition, the automatic assignment of asterisks to columns 1 and 80 started by the BOX pseudo instruction will be terminated. One blank line will be listed after the line of asterisks.

Example:

```
          | 11
 /_____|_____
/         |
|         | EBOX
|         |
(         |
```
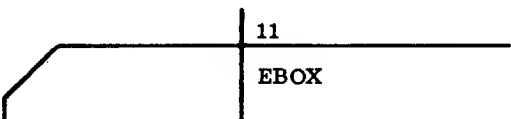
5.2.6 LST

This pseudo instruction causes the source listing to be resumed after an NLS has suspended it.

Example:

```
          | 11
 /_____|_____
/         | LST
|         |
|         |
(         |
```
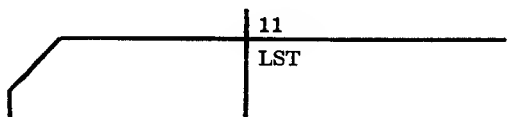
5.2.7 NLS

This pseudo instruction causes the source listing to be suppressed.

Example:

```
                              11
                              NLS
```
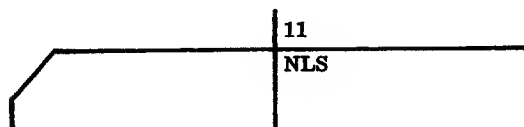
5.3 MEMORY MANAGEMENT AND SYMBOL DEFINITION PSEUDO INSTRUCTIONS

These pseudo instructions define symbols and provide for controlling the allocation of micro memory for the object code output. In addition, two memory management maps can be produced that list zero locations and locations set by ORG pseudo instructions.

Both the EQU and ORG pseudo instructions require an address expression that begins in card column 17 and may continue through column 80. The expressions are made up of operands separated by operators. The operands may be constants or previously defined symbols. The operators are +, -, *, and / (add,

subtract, multiply, and divide). Expressions are evaluated from left to right; operators are executed as they are decoded. Parentheses are not allowed to group operators within an expression. The expression terminates on the first blank character. The range of the value of the expression is from 0 to FFF_{16}. Any constant used in the expression also has the same range.

Example:

A+B-C*D/E	Legal (if A = 5, B = 2, C = 3, D = 4, E = 5, value is 3)
F	Legal
-1	Illegal (out of range)
A+-B-C*D/E	Illegal (two operators without an operand in between)

5.3.1 EQU

The EQU pseudo instruction assigns a value corresponding to the expression beginning in column 17 to the symbol appearing in the location label field (columns 2 through 9). The symbol in the location field takes on an upper quality if the expression has more than one term in it. If the expression consists only of a single, defined symbol, the symbol in the location field takes on an upper or lower quality matching that of the symbol in the A field. The use of the plus sign or the minus sign in column 1 of an EQU card has no effect on the quality of the symbol defined by the equate operation.

The EQU processing takes place during pass 1 of the assembler, and any symbol appearing as an operand in the expression must have appeared and been defined in a location field prior to its use in the EQU pseudo instruction.

Example:

2	11	17	
VALUE	EQU	4	Set VALUE = 4
LABEL	EQU	VALUE * VALUE/2	Set LABEL = 8
LABEL.1	EQU	LABEL + 1 * 2	Set LABEL.1 = 18
P.43X	EQU	VALUE/3	Set P.43X = 1

5.3.2 ORG

The ORG pseudo instruction is used to assign a starting value to the micro-memory allocation counter. The micro-memory allocation counter provides for automatic allocation of micro instructions to successive upper and lower locations, unless the allocation is changed by the coding of a plus sign or minus sign in column 1 of the micro-instruction input card. When ORG is encountered, all instructions and data following the ORG pseudo instruction are assembled in consecutive upper and lower micro-memory locations, starting with the upper location of the address specified by the expression beginning in column 17. The ORG may be used as many times as desired. If use of the ORG pseudo instruction

causes some instruction to be assembled into a non-zero instruction (i.e., assembly over an already assembled location), an error is flagged and the number of the card that previously caused the location to be assembled is printed for cross-reference. The most recent instruction does, however, overlay the previously assembled instruction.

The line of micro code, or constants, following the ORG instruction is assembled as an upper instruction unless the instruction assignment is overridden by a minus sign (-) in column 1 of the instruction following the ORG pseudo instruction.

Examples:

11	17	
ORG	100 + ABC	Set program location counter to upper of 150 decimal if ABC = 50
ORG	FF3X	Set program location counter to upper of $FF3_{16}$
ORG	TAG	Set program location counter to the upper of location value TAG (provided TAG is defined)

5.3.3 USE OF QUALIFIER FIELD

The qualifier field of a micro-instruction input card (column 1) also controls the operation of the micro-memory allocation counter. Although not strictly a pseudo instruction, it should be mentioned.

As each micro instruction is assembled, the micro-memory allocation counter is increased by a half micro-memory word in preparation for the assignment of the next micro instruction or constant. The qualifier field operates in adjusting the micro-memory allocation to meet the programmer's desires.

5.3.3.1 PLUS QUALIFIER

If the micro-memory allocation counter has advanced to assign the current micro instruction to an upper micro-memory location, the plus qualifier has no effect. If the micro-memory allocation counter has advanced to assign the current micro instruction to a lower micro-memory location, that location will be left zero and the micro-memory allocation counter will be advanced to the upper location of the next micro-memory address.

5.3.3.2 MINUS QUALIFIER

If the micro-memory allocation counter has advanced to assign the current micro instruction to an upper micro-memory location, the minus qualifier will cause that location to be left zero and will advance the counter to assign the current micro instruction to the lower location of that micro-memory address. If the micro-memory allocation counter has advanced to assign the current micro instruction to a lower micro-memory location, the qualifier has no effect.

5.3.4 ZMAP (ZERO MAP)

This pseudo instruction directs the assembler to produce a map of all unused (zero) locations between 0 and the highest address assembled. This pseudo instruction may appear anywhere within a program. The map will be produced after the assembly listing is complete, and will be printed on the same device used to print the assembly listing. The map includes an upper/lower flag and an address for the first zero location in a group. If there is more than one sequential zero, the number of zeros is printed (in decimal). If the number of zeros is greater than nine, the number is also printed in hexadecimal.

Example:

```
                      | 11
      _____/ |_____
    /                  |
   /                   | ZMAP
  |                    |
```
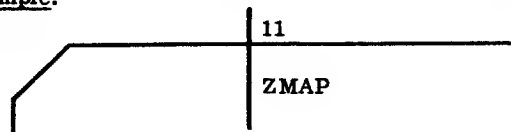
5.3.5 PMAP (ORIGIN MAP)

The PMAP pseudo instruction may be used to produce an origin map. This pseudo instruction may appear anywhere within a program. The map will be produced after the assembly listing (and the zero map if it was requested), and will be printed on the same device. The map is printed in ascending address order and includes an upper/lower flag and an address for the first instruction following each ORG pseudo instruction, as well as the card number of the instruction. The origin map is useful when trying to find the code corresponding to particular assembled locations in micro memory when the program is large and many ORG pseudo instructions are used.

Example:

```
                      | 11
      _____/ |_____
    /                  |
   /                   | PMAP
  |                    |
```
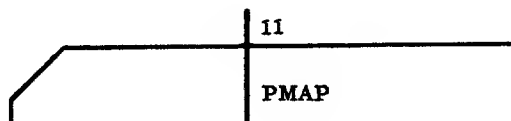
5.4 DATA DEFINITION PSEUDO INSTRUCTIONS

Three data definition pseudo instructions are provided so the programmer can define 32-bit constants to be inserted in the micro memory at the current location specified by the micro-memory allocation counter. The pseudo commands are DEC, OCT, and HEX for decimal, octal, and hexadecimal constant generation. The pseudo commands are coded in the F field of the coding form, and a string of digits in the number base is included in columns 17 through 28. Comments may start in any column after 29. The string of digits may include a minus sign (-). Embedded blanks are ignored. The string of digits is converted in its number base to a 32-bit binary number. The result is complemented if a minus sign exists in the string. A symbol may be assigned to the location label field to locate the constant, and the qualifier field may have a plus, a minus, or a blank to control the micro-memory allocation.

An error is indicated if the string of digits contains any digit not in the number base.

5.4.1 OCT

The OCT pseudo instruction causes the string of digits starting in column 17 to be converted from octal representation to binary and stored at the current micro-memory location. A symbol in the location label field is optional. The qualifier field may be used. Occurrence of any character other than 0 through 7, minus, or blank in the string will cause an error to be indicated.

Example:

11	17	
OCT	123	Create 00000000123 in the current location
OCT	-123	Create 37777777654 in the current location
OCT	1-23	Create 37777777654 in the current location

5.4.2 DEC

The DEC pseudo instruction causes the string of decimal digits in columns 17 through 28 to be converted from decimal representation to binary and stored as a 32-bit number in the current micro-memory location. A symbol in the location label field is optional. The qualifier field may be used to specify upper or lower micro-memory location. Occurrence of any character other than 0 through 9, minus, or blank in the string will cause an error to be indicated.

Example:

11	17	
DEC	10	Create 0000000A (hex) in the current location
DEC	-10	Create FFFFFFF5 (hex) in the current location

5.4.3 HEX

The HEX pseudo instruction causes the string of hexadecimal digits in columns 17 through 28 to be converted from hexadecimal to binary representation and stored as a 32-bit number in the current micro-memory location. A symbol in the location label field is optional. The qualifier field may be used to specify upper or lower micro-memory location. Occurrence of any character other than 0 through 9, A through F, minus, or blank in the string will cause an error to be indicated.

Example:

	11	17	
	HEX	DEAD	Create 0000DEAD (hex) in the current location
	HEX	DEAD-	Create FFFF2152 (hex) in the current location

5.5 PROGRAMMING INFORMATION PSEUDO INSTRUCTIONS

The programming information pseudo instructions provide the programmer with additional information in the output listing.

5.5.1 TIMING INFORMATION

The assembler analyzes each micro instruction for its execution time in the variable cycle length of the micro processor. This timing information is printed immediately preceding the first column of the card listing on the assembler printout. This timing is indicated as a blank for an A cycle, and by the letters B, C, D, E, F, and G for the corresponding cycles. The timing of the instructions is dependent on whether the micro processor is operating in ones or twos complement operation. The following pseudo instructions allow the programmer to notify the assembler of the mode of operation for timing purposes. If no timing pseudo instructions are used, the assembler assumes twos complement operation.

In addition, some instructions take a different amount of time to execute, depending on whether they are executed on a 16- or 32-bit machine. When the assembler detects a difference, two timing digits are printed on the output listing; the first is for a 16-bit machine, the second is for a 32-bit machine.

5.5.1.1 CMP1

The CMP1 pseudo instruction causes the timing information following the pseudo instruction to be listed for each instruction as if operating in the ones complement mode.

5.5.1.2 CMP2

The CMP2 pseudo instruction causes the timing information following the pseudo instruction to be listed for each instruction as if operating in the twos complement mode.

5.6 OBJECT CODE OUTPUT PSEUDO INSTRUCTIONS

The assembler creates a complete image of micro memory in the host computer during the assembly process. The assembler provides two formats for output of micro memory object code data:

- Relocatable binary card images

- Deadstart card images

In both cases, output will begin at micro-memory location 0 and continue through the address of the highest micro instruction assembled.

The assembler may produce a checksum that is included in the micro-memory image itself. The checksum feature allows the micro program to checksum itself to be sure that is was properly loaded into micro memory. It also allows the program to be sure that it has not been altered in micro memory during operation. Caution must be exercised in using the checksum feature if the program changes micro memory during the course of normal operation, since the checksum generated by the assembler cannot include the data modified in micro memory during program execution. The checksum is calculated by the assembler as follows:

$$CHKSUM = -\sum_{K=1}^{N} M_k$$

Where: CHKSUM is the calculated 16-bit, twos complement checksum

N is twice the number of micro instructions assembled (micro instructions are 32 bits each, but the checksum is 16 bits long).

$\sum$ indicates a twos complement sum

M_k is a 16-bit data item that is half of a 32-bit micro instruction.

The checksum is generated after the END pseudo instruction is read from the input stream if the checksum was requested on the object code output pseudo request card. The checksum is stored into the lower 16 bits of the micro instruction address specified on the request card if that location is zero after the END pseudo is read. If the location is not 0, the checksum is not stored and a diagnostic is produced. The requested object code output is produced in any case.

5.6.1 RELO

This pseudo instruction may appear anywhere in the micro-program source. It causes the assembler to produce relocatable binary output that is compatible with the CYBER 18 loader. (See the MSOS Reference Manual). The name of the program punched in the NAM output block is found in columns 17 through 22 of the IDENT card. In addition, program identification material is also punched in the NAM block and found in columns 23 through 66 of the IDENT card. If no IDENT card exists, the program is given a blank name.

Program-relocatable RBD blocks are punched until the entire micro memory image has been output.

An XFR block is punched as the last record of the relocatable output. The transfer address is defined by an ENT pseudo instruction. A checksum may be requested by punching an address expression starting in column 17 of the RELO card. Address expressions were described earlier in this section. The value of the address expression is the micro memory location where the checksum will be stored. The upper half of the location is used unless a minus is punched in column 1, in which case the lower half is used. The specified address must be within the bounds of the micro program and it must contain 0, or else a diagnostic results. If a diagnostic is printed, the object code output is produced, but it contains no checksum.

Example:

1	11	17
-	RELO	14X
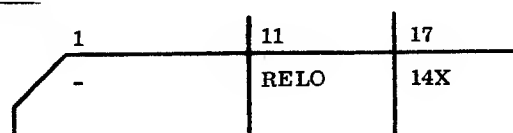
The checksum will be stored in the lower half of micro-memory address 14 (hexadecimal).

5.6.2 ENT

This pseudo instruction defines an entry point name and a transfer address to be used when producing a relocatable binary output image with the RELO pseudo instruction. The entry point and transfer name begin in column 17 and have a maximum of six characters. The value of the symbol is automatically set to zero, even though it need not be defined in the location field of a micro instruction. If the symbol is defined in the program as a value other than zero, it is still considered to be zero for the purpose of producing an entry point block and a transfer block as the result of using the RELO pseudo instruction.

Example:

```
            |11       |17
         ___|_____|_____
        /   |         |
       /    | ENT     | START
      /     |         |
     |      |         |
```
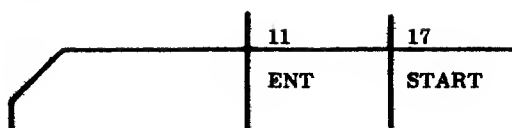
START is the entry point name associated with the micro program when the RELO pseudo instruction is used. The entry address is 0.

5.6.3 DEAD

This pseudo instruction may appear anywhere in the micro-program source. It causes the assembler to produce a set of 80-character card image records suitable for deadstarting into micro memory from a device capable of reading the card images and transferring the data to the panel interface on the micro processor. All control character strings necessary to cause the panel interface to load the data at micro-memory location 0 are embedded in the micro-memory image data. To afford proper addressing of micro memory, these control character strings also increment the N register of the micro processor by one after each block of 256 32-bit micro instructions (one-half page) has been input.

The last card punched contains code that will cause the panel interface to clear status mode register 2 and thus terminate the deadstart operation.

The number of spaces between characters punched on the card image records may be specified on the DEAD pseudo instruction card by an expression beginning in column 41. The value of the expression indicates the number of spaces punched between characters; values of 0 through 3 are legal. If column 41 is blank, a default of 1 is assumed. Blanks may be necessary, depending on the speed and characteristics of the deadstart device.

A checksum may be requested by punching an address expression starting in column 17 of the DEAD card.

Example:

```
   1          |11       |17       |41
   _____|_____|_____|_____
  /    -      | DEAD    | 1AX     | 3
 /           |         |         |
|            |         |         |
```
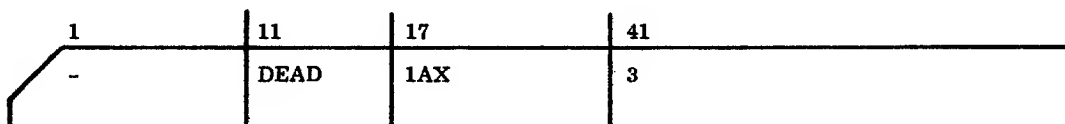
A deadstart object deck will be output on the object output device. A checksum will be produced and stored at the lower half of address 26_{10}. Three spaces will be output between each nonblank character punched.

ALU AND
A/Q SHIFT AND SCALE OPERATIONS

The F, A, B, D, and occasionally S fields of the micro instructions are used to specify operations on the arithmetic and logical unit. The F field specifies the operation to be performed. In the case of ALU operations, the A field specifies one source of operands, the B field specifies the other source, and the D field specifies the destination of the output of the ALU.

In case the F field specifies a shift of the A or A/Q register, the A, B, and D fields are not filled in on input since the assembler provides the correct values in these fields.

6.1 ALU OPERATIONS

The ALU operations are either logical or arithmetic, and combine two source inputs. The results is routed to a single destination. The two inputs are called the A source and the B source. The A source is referred to as the A input or selector 1 (S1) and the B source as the B input or selector 2 (S2).

6.1.1 LOGICAL OPERATIONS

The logical operations perform bit-by-bit combinations of the A input and B input for delivery to the destination.

An example of the use of the logical operations is shown in figure 6-1.

6.1.2 ARITHMETIC OPERATIONS

The arithmetic operations are performed in ones or two complement arithmetic and can operate on either single-precision operands, using the main ALU; or double-precision operands, using the double-precision hardware. Each arithmetic operator placed in the F field has two optional modifiers that can force a carry-input on the operation and capture the overflow condition in the status/mode register. These modifiers are used to emulate multiple-precision arithmetic and to test equalities and inequalities.

CARD VALUE	T	P/MA	MICR8-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS
7					LOGICAL OPERATIONS EXAMPLE								
9	8	800	5326 0008		EOR	A	0	X				X=(Q) EOR (A)	
10	1	800	40DE 2800		-A	X		·X				COMPLEMENT X	
11	8	881	4AE3 0008		-B		Q	Q				COMPLEMENT Q	
CARD VALUE	T	P/MA	MICR8-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS

Figure 6-1. An Example of a Logical Operation

The overflow condition exists if the signs of the A source and B source are equal and the sign of the result is different (addition), or if the signs of the A and B sources differ and the sign of the result is the same as the B source (subtraction).

An example of an overflow condition is shown in figure 6-2.

6.1.3 DOUBLE-PRECISION ARITHMETIC

The double-precision (DP) arithmetic module provides the capability to perform arithmetic on operands twice the length of the standard word size. The DP module contains three registers, A*, X*, and Q*, and an ALU (called ALU*) that is distinct from the main ALU of the CYBER 18. The A* and X* are unconditionally input to the ALU*. The output of the ALU* can be shifted left or right and the output goes to the A* register. The X* and Q* registers are loadable only; they are not destinations of the ALU*. On input to the A* register from the A source, data can be shifted right one-half word, end-around. On output from the DP registers, data can be shifted right eight bits.

An example of the double-precision operation is shown in figure 6-3.

6.2 SHIFT OPERATIONS

The shift operations in the F field specify a shift of the A register or the A/Q register. No shift is possible in the double-precision registers from this command. The N register is used in conjunction with the shift operations; the number of bits shifted is determined by the count in N at the start of the shift instruction. If the N register is zero, no shift occurs. The N register can be set in the (same) instruction by placing N = value in the C field; the value set affects the following instruction. The shift operations are various combinations of shift A or A/Q, left or right, end-around or end-off, sign-extended or not sign-extended, and entry of a 0 or 1 in the vacated bit position. The A, B, and D fields must be left blank.

An example of the shift operation is shown in figure 6-4.

6.3 SCALE OPERATIONS

Scale performs a shift operation that stops the shift when the two bits at the scale point in the A register are not equal. The scale point is normally specified as being between bits 0 and 1 in the A register.

CARD VALUE T P/NG	MICRO-NEU	LOCATION	F	A	B	D	S	C	NT	COMMENT	DIAGNOSTICS
14		•	OVERFLOW EXAMPLE								
16	1 001 7265 2000 9		ADOT	P	Q	A				OVERFLOW SAVED IN S/N REG	
18		•	NOTE THE OVERFLOW BIT IN S/N STAYS SET UNTIL IT IS EXPLICITLY CLEARED								
CARD VALUE T P/NG	MICRO-NEU	LOCATION	F	A	B	B	S	C	NT	COMMENT	DIAGNOSTICS

Figure 6-2. An Example of an Overflow Operation

CARD	VALUE	T	P/WA	MICRO-MEM	LOCATION	F	A	B	O	S	C	MT	COMMENT	DIAGNOSTICS
21					••									
22					•	DOUBLE PRECISION OPERATIONS USE THE A* REGISTER AS THE								•
23					•	LOW ORDER EXTENSION OF THE REGISTER IN THE A FIELD AND USES								•
24					•	THE X* REGISTER AS THE LOW ORDER EXTENSION OF THE REGISTER IN								•
25					•	THE B'FIELD. RESULTS ARE AUTOMATICALLY ROUTED BACK TO THE A* REG.								•
26					••									
28	0		002	5615 8A00 B		A+B	SM1	BG	SN1		0		SET DOUBLE PRECISION	
30					•	ASSUME A* HAS THE LOWER HALF OF ONE NUMBER								
31					•	ASSUME X* HAS THE LOWER HALF OF ANOTHER NUMBER								
32					•	THEN THE FOLLOWING INSTRUCTION,								
34	1		002	7110 2000 B		ADD	A	X	A					
36					•	GIVES THE RESULT THAT A,A* = A,A* + X,X*								
37					•	HOWEVER THE FOLLOWING INSTRUCTION								
39	0		003	7067 0000 B		ADD	P	Q	F					
41					•	GIVES THE RESULT THAT F,A* = P,A* + Q,X*								
CARD	VALUE	T	P/WA	MICRO-MEM	LOCATION	F	A	B	O	S	C	MT	COMMENT	DIAGNOSTICS

Figure 6-3. An Example of a Double-Precision Operation

CARD VALUE T P/MA MICRO-MEM LOCATION F   A   B   D   S   C   MT   COMMENT   DIAGNOSTICS

44   *   SHIFT EXAMPLE. ASSUME (N) IS 6 INITIALLY

46   1 003 7CD8 2000 E   AQLEA   LEFT SHIFT AQ END-AROUND

48   *   6 PLACES. (N) = FF AFTER INSTRUCTION EXECUTES.
49   *   SHIFT MOST SIGNIFICANT 8-BIT CHARACTER INTO Q AFTER CLEARINGQ

51   0 004 040B 1000   ZERO   Q   N=0   ZERO Q AND SET N TO 0
52   1 004 7CD0 2000 E   AQLEA

54   *   PLACE MOST SIGNIFICANT BIT OF A INTO LSB OF Q AND MAKE ALL OTHER
55   *   BITS 0
56   0 005 0800 101F   N=31   ASSUME HAVE 16 BIT MP
57   1 005 7040 2000 E   AQRBE

Figure 6-4. An Example of a Shift Operation

The maximum number of bits to scale is contained in the N register. On completion of the scale, the N register contains the original specified maximum minus the number of shifts necessary to position the number so the bits at the scale point are unequal. The A, B, and D fields of the coding form should be left blank so the assembler can insert the correct values.

6.3.1 SCALE EXAMPLES

The examples depicted in figure 6-5 show a scale for a 16-bit machine and a 32-bit machine. In both examples, assume that a number is positioned in the A/Q registers and has to be scaled. In the ones complement example, an end-around scale is used to provide for the propagation of the correct value for the least significant bits. In the twos complement example, a zero entry scale is used.

6.4 SELECTION OF OPERANDS FOR USE IN F FIELD OPERATIONS

The F field specifies an operation to be performed on two inputs, the A source and the B source. The result of the operation is stored in a location specified by the D field or destination. The mnemonic code specifying an A source is placed in the A field of the coding form (columns 17 through 22). The mnemonic code specifying the B source is placed in the B field of the coding form (columns 23 through 28), and the mnemonic specifying the destination is placed in the D field of the coding form (columns 29 through 34).

In the object language output, the A, B, and D fields occupy three bits each and thus allow only for specifying one of eight different sources and destinations. Since more than eight sources and destinations may be specified in each field, the S field is used to provide alternate coding interpretation for the 3-bit number in the A, B, and D fields. The CYBER 18 assembler accepts any of the specified alternate mnemonics for the fields and provides an automatic S-field setting in the object code output. The prime code set requires the setting of the S field. The result of the use of the S field to specify alternate decodings of the A, B, and D fields leads to a possible conflict of mnemonics. The resolution of this conflict is described in section 10 of this manual. The assembler also provides diagnostic messages if any conflict occurs on a programmer's input.

6.4.1 A-FIELD OPERANDS

The A field in columns 17 through 23 of the assembler coding form is used to record the mnemonic to specify the A input to the ALU. This operand may have up to two concurrent functional usages. The A-field mnemonic may:

- Specify an operand that will be functionally combined with the B source
- Specify an operand that will be supplied as the output of selector 1 for transfer to some register in the CYBER 18 system.

These uses of the A input will be covered in the examples in this section.

CARD VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	NT	COMMENT	DIAGNOSTICS
60				•		SCALE EXAMPLE ONES COMPLEMENT ARITHMETIC 16 BIT MP							
62	0	006	D0D0 1020								N=32	SET MAXIMUM SHIFT	
63	1	006	7ED0 2000 E		SDLEA							N=32-NUMBER OF SHIFTS	
65				•		SCALE EXAMPLE TWOS COMPLEMENT ARITHMETIC 32 BIT MP							
67	0	007	D0D8 1040								N=64	SET MAXIMUM SHIFT	
68	1	007	7FC8 2000 E		SDLOE							N=64-NUMBER OF SHIFTS	
CARD VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	NT	COMMENT	DIAGNOSTICS

Figure 6-5. An Example of Scale Operation

There are two groups of A-input mnemonics:

- A inputs

- A' inputs

The A inputs do not use the S field for specifying the coding, while the A' inputs require the use of a special code in the S field. The programmer should not include an S-field code if he uses an A' code because the assembler will choose the correct S field, even if a D' field is also coded.

6.4.2 B-FIELD OPERANDS

The B field in columns 23 through 28 of the assembler coding form is used to record the mnemonic to specify the B input to the ALU. Depending on the mnemonic used, the B field has two functional uses. It may:

- Specify an operand to be functionally combined with the A source

- Specify the referencing of an operand from the micro memory.

There are two groups of B-input mnemonics:

- B inputs

- B' inputs

The B inputs do not use the S field for specifying the coding, while the B' inputs require the use of a code in the S field. The assembler provides the correct coding in the S field.

In the case of the B-input mnemonics of ZERO, N, K, and NK, two bits in the C field are used as extensions of the B-field mnemonic. The assembler provides for generating the correct bits in the C field and will also allow coding other information in the C field provided the information agrees with the B-field required bits.

6.4.3 D-FIELD OPERANDS

The D field in columns 29 through 34 of the assembler coding form is used to record the mnemonic to specify the destination of information from the main organization of the CYBER 18. There are four sources of information for delivery to the specified destination. These are:

- The optionally shifted output of the ALU. This shifting occurs in a shifting network (selector 3) that provides the shift on the output of the ALU.

- The direct (unshiftable) output of the ALU

- The output of selector 1 (input to the selector is specified by the A field)

- The output of selector 2 (input to the selector is specified by the B field)

The destinations for information from the sources is indicated by the D-field mnemonic.

There are four groups of D-field destination mnemonics:

- D codes
- D' codes
- D" codes
- DD" codes

The D codes do not use the S field for specifying the coding, while the rest of the codes require the use of a code in the S field. The assembler provides the correct coding in the S field.

If an A' mnemonic is specified and a D' is specified, the assembler provides the correct code in the S field for this combination of alternate codes for the A and D fields.

The programmer should not include an S-field coding if the primed inputs are selected. (The assembler will flag an error.)

6.4.4 EXAMPLES OF USE OF F, A, B, AND D FIELDS

The assembler output listing shown in figure 6-6 demonstrates basic use of the fields discussed in this section. In some cases, the S and C fields will also be used to demonstrate common programming errors that are detected by the assembler.

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS
71					*	F,A,B,D FIELD EXAMPLES								
73		0	008	7125 0000 B		ADD	A	Q	A				A= (A) + (Q)	
75		1	000	0E9E 2703 B		A	SM2		X		K=3		X= (S/M REG 2)	
76					*	ALSO SET THE K REGISTER TO CONTAIN THE NUMBER 3								
78		0	009	F1B4 0006 B		ADD	F1	F1	F1		K=6		(F1)6 = 2*(F1)3	
79		1	009	7051 280F B		ADD	P	8G	P	15			P=(P)+1 FOR 16 BIT MP	
00		0	00A	7448 800C B		ADD+	P	ZERO	P				P=(P)+1 FOR 16 BIT MP	
81		1	00A	471A 2000		ONE	A		I				I=(A) F FIELD HAS NO EFF.	
82		0	00B	5EDF 0800				X	Q*				Q*=(X)	
84		1	008	711D 2100 B		ADD	A	X	AA*					
05					*	A*= INITIAL (A), A = (A) + (X)								
87		0	00C	54E6 0000		B		Q	X				X=(Q)	
08		1	08C	5C06 2003		A.B	X	8G	X	3			X= BIT 3 OF Q	
89		0	88D	5615 0A01 B		A+B	SM1	BG	SM1	1			SETS SM1, BIT 2	
98		1	08D	5A97 2A04 B		A.-B	SM2	BG	SM2	4			CLEAR SM2 BIT 4	
92					*	FOLLOWING CODE IS IN ERROR								
94		0	08E	700D 0700 C		ADD	SM1	CRTJ	A				A AND B BOTH USE S	16
95		1	00E	F100 2000 B		ADD.	A	M	A		K=31		C FIELD USED BY B FIELD	7
CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS

Figure 6-6. Example of the Use of F, A, B, and D Fields

The micro memory of the CYBER 18 consists of up to 4096 micro memory locations where each location is a 64-bit word that contains two micro instructions. The two micro instructions in a micro memory location are referred to as the upper and the lower micro instructions, or a micro instruction pair. Thus, a fully expanded micro memory has the capacity of storing 8192 micro instructions.

For addressing purposes, the 4096-location capacity of the micro memory is organized into 16 pages, where each page has 256 locations (or micro instruction pairs) giving a capacity of 512 micro instructions per page.

Each micro instruction is provided with an M field (coded in column 50 of the assembler coding form) that specifies the location of the next micro instruction pair from which the next micro instruction will be selected. The selection of the desired micro instruction of the next pair is determined by the coding in the T field (columns 51 through 55 of the assembler coding form).

The M field specifies the mode of obtaining the next instruction pair from micro memory; e.g., jump, return, or sequential to the next micro instruction pair. The T field specifies an unconditional selection of an upper or lower instruction from the next pair, an unconditional selection of the lower of the current micro instruction pair, or a conditional branch taking either the upper or the lower micro instruction of the next pair, depending on the condition.

7.1 M FIELD

The M field selects the next instruction pair and is coded in column 50 of the assembler input coding form. The method of selection is described below:

Code	Operation
S	Sequential addressing. Select the next micro instruction pair as the next sequential pair from the current pair. The next micro-instruction pair is within the current page.
J	Jump addressing. Select the next instruction pair from the location specified in the C field. If the jump address is in the current page, a within-page jump will be performed and the S field is available for coding. If the jump address is in another page, the micro assembler will use the S field to specify the page in a page-jump instruction.

Code	Operation
R	Return addressing. The micro memory address of the next instruction pair is obtained from the RTJ register. The top four bits specify the page and the bottom eight bits specify the address within the page. The RTJ register must have been previously set up by correct coding in the S field.
blank	The CYBER 18 assembler assumes the S-mode addressing.

The code selected in the M field may be overridden in two cases:

- If the T field has the code *L, the lower micro instruction of the current micro instruction pair is selected regardless of the M-field code.

- If the C field contains the mnemonics specifying TMA/, TMAK/, GITMAK/, or GITMAK/XT, then the transform addressing scheme replaces the M-field code.

7.2 T FIELD

The T field is located in columns 51 through 55 of the assembler coding form. This field is used to select the upper or lower micro instruction from the next instruction pair to execute. This selection may be unconditional or may be conditioned on the ALU output, value of bits in registers, reject conditions, etc.

When micro memory is being read or written as an operand, the T field is used to address the referenced micro-memory location and the upper instruction in the next sequential micro instruction pair is always selected.

The T codes may be used on all micro instructions. The T' codes are not available for use in micro instructions that have a J in the M field (jump instructions) or for micro instructions that specify N = value or K = value in the C field.

7.3 ASSEMBLER PROCESSING OF M AND T FIELDS

If the M and T fields are left blank, the assembler will assume sequential addressing mode and will choose a T code in the object code output to cause the next micro instruction to be taken as the next sequential micro instruction.

Thus, if the current micro instruction is an upper, the *L code will be inserted for the T field. If the current micro instruction is a lower, a U code will be inserted in the T field.

If a J is coded in the M field, the C field is interpreted as the location to be jumped to. The C field may contain a symbol or it may contain a constant. A symbol is carried as a total micro-memory location address and has an upper or lower property as well. A constant is interpreted as an upper of a total micro memory location address.

The assembler compares the page of the location to jump to with the page of the current micro instruction. If the page numbers are the same, a within-page jump is coded, and the S field may be used for additional instructions. If the pages are different, a page jump is coded, and the page number is extracted from the constant or symbol value and inserted in the S field for the object code. The location within page is coded in the C field of the object code. If the programmer has used a value in the S field and a page jump is coded, a diagnostic will be generated.

7.3.1 SEQUENTIAL ADDRESSING

Sequential addressing is automatically generated by the assembler if the M and T fields are blank, or the programmer may specify the addressing. The example in figure 7-1 shows assembly output of two sequences of code to show two ways of specifying sequencing. The arrows in the diagram show the program flow.

In figure 7-2, the instructions with NOP coded in the D field are not executed, but the other instructions with coding are executed. The arrows again show the program flow. This example shows how it is possible to interleave two paths of program flow through one set of micro memory locations. An alternate program could use the locations specified by the NOP in the S field.

7.3.2 JUMP ADDRESSING

In jump addressing, if no T-field value is specified, the assembler selects the T-field value to get to the instruction addressed. However, the default T-field selection is suppressed if the programmer specifies a value in the T field.

The example in figure 7-3 shows four methods of arriving at a specific micro instruction.

7.3.3 RETURN ADDRESSING

Return addressing causes control to be returned to the micro-instruction pair specified by the contents of the RTJ register. The programmer must specify a value in the T field to get a correct return location (upper or lower of the micro instruction pair). The RTJ register may be set any time by placing the mnemonic RTJ in the S field of a micro instruction. The address stored into the RTJ register is that of the next sequential micro-memory word following the instruction with RTJ in the S field. Both page and address within a page are stored in the RTJ register.

The example in figure 7-4 shows use of return addressing. In this example, a jump is made to the routine SUB which tests the value in the A register and returns to the lower of the following micro-instruction pair if the value is negative, and returns to the upper of the pair if the value in A is positive.

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	NT	COMMENT	DIAGNOSTICS
90					•								ASSEMBLER GENERATED SEQUENTIAL ADDRESSING	
100	0	00F	5F1E 0000	+		A	A					X		
101	1	00F	54E5 2000	-		B			Q			A		
102	0	010	5E0B 0000	+		A	X					Q		
104					•								PROGRAMMER SEQUENCE. NOTE SAME MICRO CODE GENERATED.	
106	0	011	5F1E 0000	+		A	A					X	S*L	
107	1	011	54E5 2000	-		B			Q			A	SU	
108	0	012	5FDB 0000	+		A	X					Q	S*L	

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	NT	COMMENT	DIAGNOSTICS

Figure 7-1. An Example of an Assembler-Generated Sequential Addressing

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	O	S	C	MT	COMMENT	DIAGNOSTICS
111					•	FURTHER SEQUENTIAL ADDRESSING								
113	0		013	5F1E 2000	+	A	A			X		SU		
114	1		013	50D8 2000	-						NOP	SL		
115	0		014	54E5 4000	+	B		Q		A		SL		
116	1		014	50D8 2000	-						NOP			
117	0		015	58D8 0000	+						NOP			
118	1		015	5ED8 2000	-	A	X			Q		SU		
119	0		016	5ADE 0000	+	ZERO				X		•L		
CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	O	S	C	MT	COMMENT	DIAGNOSTICS

Figure 7-2. Further Examples of Sequential Addressing

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	O	S	C		MT	COMMENT		DIAGNOSTICS
122					*		JUMP ADDRESSING									
124		1	016	9806 2019							LOCA	J				
125		0	017	9806 2019							LOCAA	J				
126		1	017	9A00 2019							LOCA	JU				
127		0	010	90D6 2019							LOCB	JU				
128		0	019	5806 0090	*LOCA						NOP					
129		1	019	5806 2000	-LOCB						NOP					
130		0	01A	9806 4019							LOCB	J				
131		1	01A	9806 4019							LOCBB	J				
132		0	018	9806 4019							LOCO	JL				
133		1	018	9806 4019							LOCA	JL				
134	019				LOCAA	EQU	LOCA									
135	019				LOCBO	EQU	LOCB									

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	O	S	C	MT	COMMENT	DIAGNOSTICS

Figure 7-3. An Example of Jump Addressing

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS
138					*	RETURN ADDRESSING MODE								
140	0	81C	98D8 2E58		*					RTJ	SUB	J		
141	1	81C	58D8 2000		-									
142	0	01D	58D8 0000		*					NOP			RETURN LOCATION UPPER	
143	1	01D	5AD8 2000		-					NOP			RETURN LOCATION LOWER	
144	0	01E	5F18 C000	C	SUB	A	A		NOP			SNU	SUBROUTINE TEST (A)	
145	0	01F	18D8 4AAA		*							RL	RETURN LOWER(A-)	
146	1	01F	18D8 2000		-							RU	RETURN UPPER(A+)	
CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS

Figure 7-4. An Example of Return Addressing

7.4 M- AND T-FIELD EXAMPLES

An example of the M- and T-field use is shown in figure 7-5.

Subtract the X register from the P register; if the results are negative, add (X) to (A) and place the results in the X register. If the results are positive, add (X) to (X) and place the results in the X register.

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS
149					•	M AND T FIELD EXAMPLE								
151	0	020	6858 C000	D •	SUB	P	X	NOP			SNU		
152	0	021	711E 2000	9 •	ADD	A	X	X			SU		
153	1	021	70DE 2000	9 -	ADD	X	X	X					
154	0	022	5800 0008	•					NOP			NEXT INSTRUCTION	

Figure 7-5. An Example of the M and T Fields

The special field (S field) is coded in columns 35 through 40 of the assembly coding form. If this field is not used by the assembler to specify alternate translations for the A, B, or D fields, it may be used by the programmer to specify a special instruction, it may contain a constant, or it may contain a programmer-defined symbol.

8.1 S-FIELD MNEMONICS

The mnemonics specifying alternate A-, B-, or D-field codings are not normally used by the programmer and are automatically generated as required by the assembler.

8.2 EXAMPLES

The code shown in figure 8-1 counts the number of 1 bits in the register. If a 16-bit micro processor is used, with twos complement arithmetic, the total of the number of 1 bits ends in the Q register.

The code at HERE adds X to itself to get a left shift of 1, and the COL in the T field checks for carry-out of the high order bit if it was a 1. If there is no carry-out, the upper instruction at HERE is repeated. If there is a carry-out, the lower instruction is performed, which adds one to the A register and jumps back to HERE. Each execution of the instruction at HERE counts N down by one. When N is counted down to zero, control goes to the next sequential upper instruction since a twos complement adder is assumed that would leave all zeros in the X register after the 16 additions of X to itself.

CARD	VALUE	T	P/HA	MICRO-MEM	LOCATION	F	A	B	D	S	C		HT	COMMENT	DIAGNOSTICS
157					*		S FIELD REPEAT EXAMPLE								
159	1	022	D8D8 3818		ZERO		BG	Q		N=16		SU	CLEAR Q, SET RPT COUNT	
160	0	023	54D5 200F		8		BG	A		15		SU	1 TO A	
161	0	024	70DE C200	D *HERE	ADD	X	X	X	RPT			SCOL	TEST MSB OF A	
162	1	024	8123 2024	A -	ADD	A	Q	Q		HERE		J	COUNT A 1 BIT	
163	0	025	5808 8888	*					NOP				NEXT INSTRUCTION	

CARD	VALUE	T	P/HA	MICRO-MEM	LOCATION	F	A	B	D	S	C		HT	COMMENT	DIAGNOSTICS

Figure 8-1. An Example of the S-Field Coding

C FIELD 9

The C field (columns 41 through 49 of the assembler coding form) can be used to specify the next instruction pair (if the M field is J), as a constant for setting the K or N register, as a constant for specifying the bit address for the bit generator, or it may be interpreted as additional special instructions similar to the S field.

9.1 EXAMPLE OF C-FIELD CODING

If the M field is coded with a J, the C field is used as the address of the micro-memory location to jump to. The examples in figure 9-1 show legal and illegal use of the S field in conjunction with the C field.

9.2 MULTIPLY AND DIVIDE EXAMPLES

Examples of multiply and divide codes implemented in a 16-bit micro processor using ones complement arithmetic are shown in figures 9-2 and 9-3.

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C		NT	COMMENT	DIAGNOSTICS
166					*	JUMP	SAMPLES								
168	200					ORG	200X								
169		0	200	9F1E 4501		A	A		X	LOCA	LOCN0	J		JUMP IN BANK	
170		1	200	9000 3400							LOCN2	J		JUMP OTHER BANK	
171		0	201	9F1E 2500		A	A		X	LOCA	LOCN2	J		JUMP OTHER BANK (S ERROR)	6
172		1	201	9000 2000	LOCN0						201X	JU		JUMP THIS BANK	
173		0	202	9000 3400							400X	JU		JUMP TO 0 PAGE 4	
174		1	202	9000 2000						NOP	400X	JU		JUMP TO PAGE 4 (S ERROR)	6
175	400					ORG	400X							START PAGE 4	
176		0	400	5000 0000	LOCN2					NOP				INSTRUCTION IN PAGE 2	

Figure 9-1. Example of C-Field Coding

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS
170						●●●								
179						●	MULTIPLY A BY X PRODUCT TO AQ 16 BTT MP							●
180						●	MULTIPLY WORKS ON POSITIVE NUMBERS SO PROVIDE LEAD IN TO							●
181						●	CALCULATE SIGN OF NEGATIVE INPUTS AND CORRECT AT END							●
182						●	CODE IS WRITTEN FOR ONES COMPLEMENT INPUT NUMBERS							●
183						●●●								
184					CHP1								INDICATE ONES COMPLEMENT	
185	030				ORG	30X								
186		0	030	0F18 C000 C	MULT	A	A		Q		K=0	SNU	CHECK SIGN	
187		0	031	4AE3 0045	*	-B		Q	Q		TMCK		COMP Q FOR POSITIVE	
180		1	031	5ED8 C000 C		A	X					SNU	CHECK SIGN OF X	
189		0	032	490E 0044	*	-A	X		X		DECK		GET POS X AND SIGN IN K	
190		1	032	0800 300F		ZERO				A	N=15		CLEAR A SET TIMES COUNT	
191		0	033	5008 20F2							RQRDE	SLQL	MAKE FIRST STEP TEST	
192		0	034	5F10 22F2	*	A	A		A	RPT	RQRDE	LQL	MUL ITERATION LOOP	
193		1	034	7110 22F2 C	-	ADD	A	X	A	RPT	RQRDE	LQL	MUL ITERATION LOOP	
194		0	035	4110 6000		-A	A		A			SKZU	EXIT ON POS SIGN TEST RE	
195		0	036	8110 2037	*	-A	A		A		EXIT	J	POS RESULT RECOMP A	
196		1	036	8AE3 2837	-	-B		Q	Q		EXIT	J	NEG RES COMP Q	
197		0	037	5008 0000	EXIT								NEXT INSTRUCTION	
CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS

Figure 9-2. Examples of Multiply Codes

CARD	VALUE	T	P/MA	MICRO-MEM		LOCATION	F	A	B	D	S	C	MT	COMMENT	DIAGNOSTICS
199														••	•
200					•			OIVIOE AQ BY CONTEHTS OF X. USES ONES COMPLENEHT REP								•
201					•			THIS ROUTIHE MAKES OVERFLOW TEST ANO SETS BIT IN SM1 IF PRESENT								•
202					•			F REGISTER IS USEO TO CALCULATE SIGH OF QUOTIENT								•
203														••	•
204	1	037	5F1F C000	C		OIVIOE	A	A		F			SHU	CHECK SIGH	
205	0	038	4110 2000		•		-A	A		A			SU	NEG COMP A	
206	1	030	5000 4000		-						NOP		SL	POS. LEAVE ALOME	
207	0	039	4AE3 0000		•		-B		0	Q				NEG CONP Q	
208	1	039	D2EF 3000				EOR	X	F	F		N=13		QUOT SIGN IN F. SET CTR	
209	0	03A	5E0A C061	C			A	X		I		CLRF/1	SHU	SAVE X IN I, SET 2S COMP	
211							CHP2								INOICATE TWOS COMPLEMENT	
213	1	03A	400E 2000				-A	X		X				GET POSITIVE X	
215					•		CHECK FOR OVERFLOW									
217	0	039	5F10 0070				A	A		A		ROLXN		SHIFT AQ LEFT 1	
218	1	038	6910 C0F0	0			SUB	A	X	A		ROLXN	SCOL	TEST OIVIOE OVERFLOW	
219	0	03C	7110 C070	0	•		ADO	A	X	A		RQLXN	SNU		
220	1	03C	5615 2A0E	B	-		A+B	SM1	BG	SM1		14		SET BIT 14 OIVIOE OVERFLO	
222					•		OIVIOE ITERATION LOOP									
224	0	030	7110 C270	0	•		AOO	A	X	A	RPT	RQLXH	NU		
225	1	030	6910 C270	D	-		SUB	A	X	A	RPT	RQLXN	NU		
227					•		ENO CORRECTION A IS 1 LEFT ANO MAY RE 1 TO MAHY SUBTRACTS									
229	0	03E	7110 2000	B	•		AOO	A	X	A			SU		
230	1	03E	5F10 4076		-		A	A		A		QROE	SL		
231	0	03F	7110 0000	B	•		AOO	A	X	A					
233					•		CHECK SIGH OF QUOTIENT									
235	1	03F	5F50 C051	C			A	F				SETF/1	SNU	SET OHES COMP	
237							CHP1								INOICATE ONES COMPLEMENT	
239	0	040	4AE3 0000		•		-B		0	Q				COMP QUOTIENT	
241					•		CHECK SIGH OF REMAIHOER AMO CORRECT A									
243	1	040	52A0 C000	C	-		EOR	I	F				SHU	CHECK REM SIGH	
244	0	041	0110 2037		•		-A	A		A		EXIT	J	OOME	
245	1	041	9000 2037		-							EXIT	J	OOME MO CHANGE	
CARO	VALUE	T	P/MA	MICRO-MEM		LOCATION	F	A	B	D	S	C	MT	CONNENT	OIAGNOSTICS

Figure 9-3. Examples of Divide Codes

The proper selection of a mnemonic for a field depends on the mnemonics used in the other fields (table 10-1 lists permissible combinations of codes). This is an inherent characteristic of the CYBER 18 and is a result of maximizing the information content in the instruction repertoire. This means that the most frequently used operands require less space than the less frequently used operands.

TABLE 10-1. LEGAL F, A, B, D, AND S COMBINATIONS

F FIELD	A FIELD	B FIELD	D FIELD	S FIELD
Arithmetic or logical	A	B	D	Unused
	A'	B	D	AP
	A	B'	D	BP
	A	B	D'	DP
	A'	B	D'	APDP
	A	B	D''	DPP
	A	B	DD	DD
Shifts or scale	b	b	b	Unused
	b	b	NOP	Unused
Where: b is a blank field				
A, A', B, B', D, D', DD are types of A-, B-, and D-field codes				
AP, BP, DP, APDP, DPP, DD are values supplied by the assembler				

Each instruction also consists of a maximum of four independent and concurrently executed functional operations; this further reduces the effective instruction execution time. Their associated fields are also dual, in the sense that they can be used to specify less frequently used operands for other fields.

The instructions are further enhanced through a merging of firmware/hardware concepts in the transform boards. The transforms provide functions that would normally require several instructions (without a transform). A transform is essentially the mapping of bits within the CYBER 18 architecture including any desired constants to the MIR, MA, N, and K registers. The transform resultant can modify both the state of the registers and the instruction sequencing. The assembler flags ambiguously coded instructions with an assembly error message. See figure 10-1.

```
CARD VALUE T P/MA  MICRO-MEM  LOCATION F     A     B     D     S     C       MT    COMMENT                                        DIAGNOSTICS

247                           ********************************************************************************
248                           *          EXAMPLES OF CONFLICTING MNEMONIC SELECTION AND ASSEMBLER ERROR CODES *
249                           *          BLANK FIELDS ARE AVAILABLE FOR USAGE                                 *
250                           *          A B D AND S FIELD CONFLICTS                                          *
251                           ********************************************************************************

252     8 042  7842 8000 C             ADO   P     F2    I     HALT                LEGAL
253     1 042  7842 2780 D             ADO   M1    F2    I     HALT                ILLEGAL                                        17
254     8 043  7842 0700 O             ADO   M1    F2    I                         LEGAL
255     1 043  5F8E 2008 B             A     A     CRTJ  X                         LEGAL
256     0 044  5F8E 0000 B             A     A     CRTJ  X     LOEA                ILLEGAL                                        17
257     1 044  5F1C 2900              A     A     X     M1                        LEGAL
258     8 045  5F1C 0900              A     A     X     M1    READ                ILLEGAL                                        17
259     1 045  5C25 2A00 0            A.0   SM1   Q     SM1                       LEGAL
260     0 046  5C25 0A00 B            A.B   SM1   Q     SM1   RTJ                 ILLEGAL                                        17

261                           ********************************************************************************
262                           *          T AND C FIELD CONFLICTS                                             *
263                           ********************************************************************************

264     1 046  D8D0 3021                                            N=33    U     LEGAL
265     0 047  D8D0 1021                                            N=33    LQL   ILLEGAL                                        19
266     1 047  58D0 2000   LAB1                                     256     LQL   LEGAL
267     0 048  98D0 4047                                            LAB1    JLQL  ILLEGAL                                        19

268                           ********************************************************************************
269                           *          B AND C FIELD CONFLICTS                                             *
270                           ********************************************************************************

271     1 048  54C0 2000                           N                   31          ILLEGAL                                        7
272     0 049  54C8 0000                           N                   108         LEGAL

273                           ********************************************************************************
274                           *          SHIFT FUNCTION AND A OR B FIELD CONFLICT                            *
275                           ********************************************************************************

276     1 049  7C90 2000 E            ALEA  A                                      ILLEGAL                                        1
277     0 04A  7C90 0000 E            ALEA        Q                                ILLEGAL                                        3
270                                   END

        13 LINES CONTAIN ERRORS

CARD VALUE T P/MA  MICRO-MEM  LOCATION F     A     B     D     S     C       MT    COMMENT                                        DIAGNOSTICS
```

Figure 10-1. Examples of Conflicting Mnemonic Selection and Assembler Error Codes

The assembler prints numeric error codes to flag and diagnose incorrect assembly statements. When a statement is in error, one to four error codes are listed to the right of the statement to describe the problem. Table 11-1 contains a list of the Micro Assembler error codes. Figure 11-1 is an example of a listing containing error codes.

TABLE 11-1. MICRO ASSEMBLER ERROR CODES

NUMERIC CODE	MEANING OR CAUSE
1	The A code is set by a shift operation in F.
2	The A code is undefined.
3	The B code is set by a shift operation in F.
4	The B code given is undefined.
5	A C- and M-field conflict occurred.
6	Cannot reach page; the S code is set and cannot be used to reach the page specified in the C field for the jump command.
7	C- and B-field conflict.
8	The C code given is undefined.
9	A multiply-defined label was encountered.
10	Not used.
11	The D code is setting an S code in conflict with A or B.
12	The D code is undefined.
13	The EQU pseudo instruction needs a symbol in the label field.
14	The F code is undefined.
15	The M code is undefined.
16	A different S code is set by the A and B fields.
17	The S code is already set by A, B, or D.
18	The S code is undefined.
19	An illegal T code was given for a jump, N=, or K=.
20	A T code is required but is not specified; it is assumed to be U.

TABLE 11-1. MICRO ASSEMBLER ERROR CODES (Continued)

NUMERIC CODE	MEANING OR CAUSE
21	The T code is undefined.
22	An undefined symbol was encountered; the field containing the symbol is specified in the next integer.
23	There is an illegal character in the HEX, DEC, or OCT constant.
24	Not used
25	The numeric value is not in the range (0 - $FFF)
26	The micro-memory location is greater than 4,095.
27	The first character on the card is not *, $, +, -, or blank.
28	The shift code in the C field is illegal when the S field contains the L8EA instruction.
29	The D field must be an NOP if: 1. The F field is a shift or scale, or 2. The B field is MMU or MML.
30	KZU in the T field is illegal if the C field is INCK.
31	NZU in the T field is illegal if the C field is INCN.
32	Macro-memory read in the C or S fields is illegal if the instruction time is E or F.
33	The A field may not be blank on an EQU card. The symbol in the L field is undefined.
34	This location has already been used. The next integer specifies the numbering of the card that caused the location to be previously assembled. This instruction overrides the one previously assembled.
35	Not used
36	The address is out of range (less than zero or greater than 4,095).
37	A symbol in an address expression is longer than eight characters.
38	The number of spaces requested between characters on the deadstart object cards is less than zero or greater than three. If this error occurs, one space will be punched between characters.

TABLE 11-1. MICRO ASSEMBLER ERROR CODES (Continued)

NUMERIC CODE	MEANING OR CAUSE
39	An error has caused the label in columns 2 through 9 of an EQU card not to be defined. The error may be one of the following: • Use of a symbol that has not been previously defined in the EQU expression. • Use of a symbol that is larger than eight characters in the EQU expression. • The value of the EQU expression is greater than 4,095 or less than zero.

In addition to the error codes in figure 11-1, there are four other conditions:

Message	Printed	Meaning
*****CHECKSUM ERROR*****	At the end of the source code listing. The error is detected in pass 2.	Contents of the address in object code output to contain twos complement checksum was not zero. The object code output is produced without a checksum.
STOP 5	CYBER 18-17 system: On list output device when the error occurs. The error is detected in pass 1. CYBER 170/70/6000 system: In dayfile.	Binary object code. The page read from mass storage was not the page requested. An error was detected in subroutine GETPAG.
STOP 444	CYBER 18-17 system: On list output device when error occurs. The error is detected in pass 1. CYBER 170/70/6000 system: In dayfile.	Symbol table overflow; more than 10,000 symbols have been defined.
STOP 777	CYBER 18-17 system: On list output device when a FINIS card is read. CYBER 170/70/6000 system: In dayfile.	This is not an error. It signifies normal job termination.

```
CARD VALUE T P/MA  NICRO-MEM  LOCATION F     A     B     D     S     C     MT  COMMENT                    DIAGNOSTICS
  1                                    IDENT OIAGNOSTIC EXAMPLE FOR THE MP ASSEMBLER

  2                                    **************************************************************************
  3                                    *                                                                        *
  4                                    *         OIAGNOSTIC EXAMPLE                                              *
  5                                    *                                                                        *
  6                                    **************************************************************************

  7  000                               TROUBLE EQU                                                 UNDEFINEO EQUATE          33
  8  000                                       EOU   5                                             UNDEFINED LABLE           13
  9  00A                                       OPG   AX                                            
 10      0  00A  5008 0000  SYM                                                                    FIRST SYMBOL OUP           9
 11      1  00A  5000 2000                                        NOP                              
 12      0  00B  5808 0000  SYN                                                                    OUPLICATE SYMBOL          9
 13      1  008  9808 200A                                              SYN   J                    USE OF OUP. SYMBOL
 14      0  00C  5000 0000  LOC                                   NOP                              GOOD STATEMENT
 15      1  00C  5008 2000           NP                                                            UNDEFINEO FUNCTION        14
 16      0  00D  5808 0000                Q                                                        CAN'T USE Q AS A   INPUT   2
 17      1  00D  5808 2000                      A                                                  CAN'T USE A AS B   INPUT   4
 18      0  00E  5808 0000                            XM1                                          UNDEFINEO OESTINATION     12
 19      1  00E  5000 2000  X                               RAP                                    UNDEFINEO S CODE          22     18
 20      0  00F  5800 0000  X                                                                      ILLEGAL CHAR IN COL 1     27
 21      1  00F  5808 2000                                              JOE                        UNDEFINEO C FIELO         22      8
 22      0  010  BFFF OFFF                                                    X                    ILLEGAL N CODE            15
 23      1  010  5808 2000                                                    BTL                  ILLEGAL T COOE            21
 24      0  011  7C80 0000 E  ALOE  X                                                              A,B MUST BE BLANK          1
 25      1  011  7C00 2000 E  ALOE        X                                                        A,B MUST BE BLANK          3
 26      0  012  5570 0700 B        A*    INTA                                                      A,B CAN'T BOTH BE PRINE   16
 27      1  012  5E10 2700 B        SN1         A*                                                  A*,O*' ILLEGAL           11
 28      0  013  5E18 0700 B        SM1               REAO                                          A' S CONFLICT            17
 29      1  013  5400 2000 B              INRO        LOEA                                          B' S CONFLICT            17
 30      0  014  5800 0800                      A*    GATEI                                         O' S CONFLICT            17
 31      1  014  54C8 2008              N                    OECK                                  B AND C CONFLICT          7
 32      0  015  0808 OFFF                                  N=35   R                               C AND N FIELO CONFLICT    5
 33      1  015  9808 2300 G                                REAO   100X   J                        NEED S FIELO FOR PAGE     6
 34      0  016  0808 100F                                        N=15   LQL                       ILLEGAL T FOR N=          19
 35      1  016  9800 200C                                        LOC    JLQL                      ILLEGAL T FOR JUMP        19
 36      0  017  9808 200C                                        LOC    JU                        LEGAL T FOR JUNP
 37  00A                               ORG   10
 38      0  00A  5650 0000             A+B   P     X     A                                          ORG OVER EXISTING CODE    34     10
 39  000                               ORG   5000                                                  ORG TO NON-EXISTANT LOC.  25     36
 40                                    ENO

     27 LINES CONTAIN ERRORS

CARD VALUE T P/MA  NICRO-MEN  LOCATION F     A     B     O     S     C     MT  CONNENT                    OIAGNOSTICS
```

Figure 11-1. Assembler Diagnostic Example

EXECUTING THE MICRO PROCESSOR MICRO ASSEMBLER UNDER MSOS 4 ON THE CYBER 18 COMPUTER

The following control cards will put the Micro Assembler into execution on an MSOS system:

Control Card	Description
*BATCH	Calls the job processor.
*JOB	
*K, Lff	ff is the logical unit of the FORTRAN line printer. The first character of the output line is treated as spacing control.
*K, Prr	rr is the logical unit of the object output device.
*K, Iss	ss is the logical unit of the source input device.
*MP	Calls the Micro Assembler. The assembler reads from the source input device (ss), and prints on the FORTRAN printer (ff). If binary output was requested, it will be punched on the object output device.
*Z	Signs off the job processor.

EXECUTING THE MICRO ASSEMBLER ON A CYBER 170/70/6000 SERIES COMPUTER

The following program call card causes execution of the Micro Assembler on the CYBER 170/70/6000. It assumes the Micro Assembler is part of the system library.

MASSEM (p1, p2, p3)   p1 is the logical file name of the file on which the micro program source resides. The default is INPUT.

p2 is the logical file name of the file on which the assembler writes the source listing. The default is OUTPUT.

p3 is the logical file name of the file on which the assembler writes the object output. The default is MP17BO.

FORMAT OF RELOCATABLE OUTPUT DATA

The binary output produced by the assembler is in a format that exactly matches the format of relocatable binary programs read by the system loader on an MSOS system for a CYBER 18-17 computer. The relocatable binary consists of four different kinds of records:

- NAM
- ENT
- RBD
- XFR

When the data is output on a CYBER 170/70/6000 Series computer, all records are sixteen 60-bit words long. A 7/9 punch, sequence number, word count, and checksum are included to make the 16 words match the punched card format of a CYBER 18-17 computer. When punching the assembler's relocatable output to cards for subsequent reading on a CYBER 18-17, care must be taken to assure that the CYBER 170/70/6000 system card punch driver does not add its own sequence numbers and checksums to those generated by the assembler.

When outputting data on the CYBER 170/70/6000, unformatted FORTRAN writes are used. The FORTRAN run-time package is used for I/O on the CYBER 170/70/6000.

When data is output on a CYBER 18 computer, the data records are of varying lengths:

- NAM   34 sixteen-bit words
- ENT   5 sixteen-bit words
- RBD   56 sixteen-bit words for all RBD blocks except the last. The last RBD block has from four to 56 words, depending on the amount of data remaining.
- XFR   4 sixteen-bit words

If the data is output to a card punch, the MSOS card punch driver will add a 7/9 punch, sequence number, word count, and a checksum to each record. However, if the output device is not a card punch, the 7/9 punch, sequence number, word count, and checksum are not output to the device.

When outputting data on the CYBER 18-17, MSOS FORMAT writes are used. The FORTRAN run-time package is not used for I/O on the CYBER 18.

FORMAT OF DEADSTART OUTPUT DECK

The deadstart deck contains both data and control character strings. Control strings are punched one string per card. Micro-memory data is punched to optimize the number of complete 32-bit instructions per card, depending on the number of spaces to be punched between characters. The cards in table B-1 are punched with the assumption that a master clear (which clears the FCR) was done just prior to setting SM204 (which causes deadstart to commence).

These deadstart control cards are designed to be acceptable in the general case. No transmission is allowed to the CDT console during deadstart to handle the general case where the baud rate of the panel during deadstart is different than that of the CDT console. After deadstart, if the user wishes to see control characters typed on the panel CDT, he should type:

J40:
J58:

When outputting data on the CYBER 170/70/6000 computer, formatted FORTRAN writes are used. The FORTRAN run-time package is used for all I/O on the CYBER 170/70/6000.

When outputting data on the CYBER 18-17 computer, MSOS FORMAT writes are used. The FORTRAN run-time package is not used for I/O on the CYBER 18-17.

Whichever computer is used, the deadstart card images are 80 characters each.

Figure B-1 is a listing of a deadstart deck that was produced by assembling the program listed in Appendix C.

TABLE B-1. DEADSTART DECK FORMAT

TYPE	CARDS	MEANING
1	K20089000G	Set up FCR register. Select K register for display 0. Select FCR for display 1. Select MICRO mode. Suppress console transmit. Enable micro-memory write.
2	L00G	Clear K register.
3	J01G	Select N register for display 0.
4	L00G	Clear N register.
5	J0CG	Select micro memory for display 0.
6	L	Begin load of micro memory.

TABLE B-1. DEADSTART DECK FORMAT (Continued)

TYPE	CARDS	MEANING
7	Data	Micro-memory data in 32-bit micro instructions. Each instruction is terminated with a G. The number of instructions per card depends on the spacing.
		Spaces Instructions per card
		0 8
		1 4
		2 2
		3 2
8	K9A088000G	Set up new FCR register. After all data has been punched: Select RTJ register in display 0. Select SM2 register in display 1. Select MICRO mode. Suppress console transmit. Disable micro-memory write.
9	K00000000G	Clear SM2. This card stops the deadstart hardware from reading more cards.

```
K 2 0 0 8 9 0 0 0 G
L 0 0 G
J 0 1 G
L 0 0 G
J 0 C G
L
9 4 D E 2 5 0 5 G      D 8 D F 3 9 0 9 G    5 4 D 5 0 9 0 9 G    D 8 D D 3 0 0 3 G
C 6 D E 0 0 F F G      5 4 E 0 4 8 8 0 G    7 1 1 D 0 0 0 0 G    5 4 D 0 E 0 1 F G
9 4 D E 2 5 0 0 G      5 4 E 8 4 8 8 0 G    7 1 1 D 6 0 4 4 G    0 0 0 0 9 2 D A G
5 8 D E 8 9 4 C G      9 8 D C 4 9 0 2 G    5 D 0 D E 0 0 0 G    8 6 D E 4 0 0 2 G
9 8 D 8 2 D 0 8 G      5 8 D 8 2 0 0 0 G
K 9 A 0 8 8 0 0 0 G
K 0 0 0 0 0 0 0 0 G
```

Figure B-1. Deadstart Deck Example

MICRO-MEMORY CHECKSUM   C

The checksum option, available with the object code output options (DEAD and RELO), is very useful when the programmer wishes to be able to initially or periodically verify that the micro program is intact in micro memory. Caution must be taken to ensure that the memory included in the checksum (all memory within the bounds of the assembled program) is not changed by normal program function. If it is changed, the checksum routine, coded as part of the user's program, will detect an error.

The code in Figure C-1 illustrates one way of coding a checksum routine that will run on either a 16-bit or a 32-bit micro processor. It is nine words long and occupies the first nine locations of micro memory. Following are some notes to help clarify its operation:

- When starting a micro processor after a master clear, the first location executed is 0 lower.

- This checksum routine requires the program to be N blocks long where a block is 128 words. There are two blocks per page and 16 pages of micro memory in the maximum configuration.

- K7L must be coded in the T field when reading micro memory to allow reference to the least significant bit of the K register to determine which half word is to be read.

- After reading micro memory, the next micro instruction executed is the upper of the next sequential location.

- Driving the bit generator with a 31 in the C field produces a zero on a 16-bit machine.

- The checksum produced by the assembler is in a 16-bit twos complement form.

- The machine is in twos complement mode (SM101 = 0) while doing the checksum addition; thus, even on a 32-bit machine, the lower 16 bits of the A register represent the correct checksum value. This is because in twos complement addition, there is no end-around carry.

- After completing the checksum loop, N and K both equal FF. This is because the test for K and N equal to zero is done before the registers are decremented. If the N zero (NZU) and K zero (KZU) T-field tests are true, N and K are still decremented (0 - 1 = FF).

- The X register is set to all ones before reading micro memory in case a hardware malfunction causes no data to be gated to X after micro memory is read.

- The definition of the checksum from section 5.6 is:

$$CHKSUM = -\sum_{K=1}^{N} M_k$$

It is clear that if this checksum value is stored into micro memory by the assembler at a location (M_k) that was 0 and was included in the checksum calculation by the assembler, then the checksum generated in the A register by the following sample code must be zero if the correct data was loaded into micro memory and the micro processor was functioning properly because:

$$A = \sum_{K=1}^{N} M_k + CHKSUM \equiv 0$$

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	O	S	C	MT	COMMENT	DIAGNOSTICS
1						IDENT CHECKSUM EXAMPLE								
2	009				MICROHLT EQU 9							ENABLE MICRO-INSTRUCTION HALT	
3	003				BLOCKS EQU 3							(NUMBER OF MICROMEMORY PAGES) * - 1	
4	01F				BGBIT31 EQU 31							BIT GENERATOR DRIVER VALUE FOR BIT 31	

```
 5          ****************************************************************************
 6          *                                                                          *
 7          *                   I N I T I A L I Z A T I O N                             *
 8          *    THE CHECKSUM IS 2'S COMPLIMENT, 16 BIT                                 *
 9          *    THE CHECKSUM IS CALCULATED AND SHOULD RESULT IN A REG = 0.             *
10          *    IF THE A REG IS NOT ZERO, THE CHECKSUM FAILS (DETECTS AND ERROR).      *
11          *    IF THE CHECKSUM FAILS, THE PROGRAM HALTS AT BADSUM UPPER.              *
12          *    IF THE CHECKSUM PASSES, THE PROGRAM CONTINUES AT BADSUM LOWER          *
13          *    WITH THE REGISTERS AS FOLLOWS                                          *
14          *                                         MP16          MP32               *
15          *                                                                          *
16          *                     REGISTER            VALUE         VALUE    N.U. = NOT USED *
17          *                        I                N.U.          N.U.     X = UNKNOWN     *
18          *                        P                N.U.          N.U.              *
19          *                        A                0000          00000008          *
20          *                        F                N.U.          N.U.              *
21          *                        X                XXXX          XXXXXXXX          *
22          *                        O                N.U.          N.U.              *
23          *                        SM1              0040          00400000          *
24          *                        M1               0000          00000000          *
25          *                        SM2              0000          00000000          *
26          *                        M2               0000          00000000          *
27          *                        N,K              FFFF          0800FFFF          *
28          *    NOTE THAT LOWER OF SECADD IS OPEN TO CONTAIN                           *
            *    THE CHECKSUM VALUE GENERATED BY THE ASSEMBLER                          *
            ****************************************************************************
```

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	O	S	C	MT	COMMENT	DIAGNOSTICS
29	8	000	940E 2505	+MP32XFIX	B			X	X	LOEA	SECADO	J		FINISH SWAPPING X REG.
30	1	800	080F 3909	-ZEROLOWR	ZERO				SM2		N=09		CLR SM2, SET N FOR HALT
31	0	081	5405 0909	+	B		BG	SM1		MICROHLT		SET SN1 UP FOR HALT ONLY
32	1	881	0800 3003	-	ZERO			A		N=BLOCKS		CLR A, SET N TO READ MM.
33	0	082	C60E 00FF	+SETKFF	ONE			X		K=FFX		CLR M2, SET K TO READ MM.
34	1	002	54E0 4880	F -SUMLOOP	B		MMU					K7L	READ MICRO MEMORY
35	8	083	7110 0008	9 +	A00	A	X	A					ADD THIS TO SUM
36	1	083	54D0 E01F	C -	B		BG			BGBIT31	ZL	ZERO IF MP16
37	0	004	940E 2500	+	B		X	X	LBEA	MP32XFIX	J		SWAP HALVES OF X REGISTER
38	1	004	54E8 4880	F -	B		MML					K7L	READ 2ND 1/2 OF WORD
39	0	005	7110 6044	9 +SECADD	A00	A.	X	A		DECK	KZU	ADD 2ND 1/2 OF WORD
40	0	006	580E 894C	+	ZERO			M2		DECN	NZU	CHECK THE M REGISTER
41	1	006	980C 4902	-	ZERO			M1		SUMLOOP	J	CONTINUE TO SUM IN BLOCK
42	0	007	500D E000	C +CMFCKSUM	A.B	A	N,K	A				ZL	CHECK THE SUM FOR ZERO
43	1	007	86DE 4002		ONE			X		SUMLOOP	J	SET X, READ NEXT WORD
44	0	008	9808 2008	+BADSUM					HALT	BADSUM	J	BAD CHECKSUM, A NOT ZERO
45	1	008	58D8 2000	-				NOP				CHECKSUM CORRECT
46					ENO									

NO ERRORS

CARD	VALUE	T	P/MA	MICRO-MEM	LOCATION	F	A	B	O	S	C	MT	COMMENT	DIAGNOSTICS

Figure C-1. Checksum Example

SAMPLE LISTING INCLUDING ORIGIN MAP AND ZERO MAP   D

The following descriptions are keyed to the fields (columns) of the sample listing:

```
                          A'
          ┌──────────────────────┐
MP MICRO ASSEMBLER 1700 VERSION 16.0   SAMPLE LISTING                    G
 A    B    C      D                                                      │
┌─┐ ┌───┐┌───┐┌────────┐ ┌────────────────────────────────────────────┐
CARD VALUE T P/MA  MICRO-MEM  LOCATION  F    A    B    D    S    C    MT   COMMENT                    DIAGNOSTICS
                                                                                                        H
  12                  E F
  13                  │ │
  14                  │ │
  15                  │ │
  16                  │ │
  17   006            ▼ ▼         ORG   3X*2+MEMREF1
  18       0  086  580F 2340 G +OVI    ZERO                    F    READ  CLRK      U    READ (EA), F=NK=0
  19       0  007  553E 0C00 +          B    A    MEM  X    GATEI           (EA) TO X, SAVE A IN I
  20       1  087  94E5 CF20 C -        B         Q    A    CLRNP OVI.10 JNU  CHECK SIGN OF Q
```

Field	Contents
A'	Assembler identification: host machine type (CYBER 18-17/CYBER 170/70/6000), assembler version number
A	Source card number (in decimal)
B	Value (in hexadecimal) of the expression on an EQU or an ORG card.
C	Micro memory instruction location (in hexadecimal) assigned to this card. The P/MA column contains three digits. The first is the page address; the second two are the micro-memory address within the page. The T column specifies the upper half word for T = 0. T = 1 specifies the lower half word.
D	The contents in hexadecimal of the instruction location T P/MA.
E	A code indicating the length of time required to execute this instruction. A blank is an A time.
F	When this column is not blank, the instruction on this card takes longer to execute on a 32-bit MP than on a 16-bit MP. The code printed in this column will indicate the execution time on a 32-bit machine (see MP engineering specification, Section 3.3.1.5).
G	Card image. The fields on the card are indicated by the notations: LOCATION, F, A, B, D, S, C, M, T, and COMMENT.
H	If the assembler detects an error in the information coded on the source card, the error code(s) is (are) printed on this part of the listing. There is room to print up to four error codes on the listing.

```
CARD VALUE T P/MA  MICRO-MEM   LOCATION F    A    B    D    S    C         MT    COMMENT                      DIAGNOSTICS

 12                                ***************************************************************************
 13                              *                                                                          *
 14                                O I V I D E   I N S T R U C T I O N              F=3                      
 15                              *                                                                          *
 16                                ***************************************************************************

 17   086                          ORG  3X*2+MEMREF1
 1A      0  086  5A0F 2340 G +DVI  ZERO           F         READ CLRK       U     READ (EA), F=NK=0
 19      0  087  553E 0C00   +      B    A    MEM  X         GATEI                 (EA) TO X, SAVE A IN I
 20      1  087  94E5 CF20 C -      B         Q    A         CLRNP DVI.10    JNU   CHECK SIGN OF 0

 22   086                          ORG  3X*2+MEMREF1
 23      1  086  591F 4C40   -      ZERO A          F        GATFI CLRK      L     EA IN X (IM. OPR.),F=NK=0
 24                              +   B         0    A         CLRNP DVI.10    JNU   SAVE A, CHECK SIGN OF Q

 26   020                          ORG  20X+CNK+PG0
 27      0  020  4AE5 0244 +DVI.10  -A        Q    A         RPT  DECK             MAKE Q POS (0 TO A) NK=-0
 28      1  020  928B 4E22   -      EOR  I    N,K  0         RTJ  DVI.30     J     COMP 16LS3 IF NECESSARY

 30                                CMP2
 31      0  021  48DE 0C00 +DVI.20  -B   MEM  X    X         GATEI                 MAKE X POSITIVE, I=FFFF
 32      1  021  02AF 3010   -      EOR  I    M,K  F              N=16             F=+0/-0 FOR OUOT SIGN
 33      0  022  555F 4C74   +      B    F    X    F         GATEI RLOE      L     SAVE F IN I, F=2*X

 35      1  022  155E CC61 C -DVI.30 B   F    X    X         GATFI CL0F/ONESRNU    CHECK SIGN OF X, I=F=+-0,
 36                              *                                                 GO TO OVI.20 VIA RTJ PEG

 38                              *   OIVIDE ALGORITHM - SUBTRACT, SHIFT AND CORRECT (IF NECESSARY). OUOTIENT IS
 39                              *   FORMEO BY THE COMPLEMENT OF THE CARRYOUT OF A (COA) BEING PUT IN THE
 40                              *.  LSB OF Q, SHIFTED, AND THE OPERATION REPEATEO 17 TIMES. IF COA IS A
 41                              *   ONE THE FIRST TWO TIMES, IT INDICATFS AN OVERFLOW. THE REMAINDER*2
 42                              *   NILL BE IN A AND NILL NEEO CORRECTEO BY X*2 IF NEGATIVE. THUS,
 43                              *   A,O / X = O, REMAINOER*2 IN A (IF NEGATIVE, R*2 - X*2)
 44                              *
 45                              *   EACH ITERATION RESULTS IN    A=A-X (OR A=A-2X+X), A,Q=A,O*2, Q00=NOT(COA)

 47      0  023  7110 C270 0 +      AOD  A    X    A         RPT  ROLXN      NU    CORRECTION(A=A-2X-X)SHIFT
 48      1  023  6910 C270 0 -      SUB  A    X    A         RPT  ROLXN      NU    SUBTRACT, (A=A-X),  SHIFT

 50      0  024  7120 0000 A +      AOD  A    F    A                               CORRECT REMAIN(R=R-2X+2X)
 51      1  024  0E9E 3001   -      A    I         X              N=1              SAVE I IN X FOR OUOT SIGN
 52      0  025  7010 0051 E        AREA                          SETF/ONES        POSITION REMAIN. PROPERLY
 53      1  025  9720 CC2A C        A+B  A    0         GATFI OVI.40        JNU    SAVE A IN I, CHECK OVF.
 54                                CMP1

 56   02A                          OPG  2AX+PG0
 57      0  02A  5615 0A0A B +DVI.40 A+B SM1  BG   SM1        OVEPFLOW              OVERFLON (1ST TWO COA=1)
 58      1  02A  52E5 6000   -      EOR  X    Q    A                         KZU   OUOT WIM PROPER SIGN TO A
 59      0  029  9E98 2058   +      A    I         Q              INI        J     REMAINOER WITH SAME SIGN
 60      1  028  809B 2058   -      -A   I         Q              INI        J     AS OIVIO. TO Q, GOTO INI

CARD VALUE T P/MA  MICRO-MEM   LOCATION F    A    B    D    S    C         MT    COMMENT                      DIAGNOSTICS
```

Figure D-1. Sample Listing

```
********* ORIGIN  MAP *********


    T P/MA          CARD


    0 0020            27
    0 002A            57
    0 0006            10
    1 0006            23
```

```
********* ZERO MAP *********


    T P/MA        NUMBER


    0 0000        64 0040
    0 0026         0
    0 002C        100 0084
```

Figure D-1. Sample Listing (Continued)

ASSEMBLER INSTALLATION E

The assembler is basically written in FORTRAN to provide transportability between the CYBER 18-17 and CYBER 170/70/6000 computers. However, some basic differences in FORTRAN as implemented on the two machines require some differences in the programs themselves. The differences have been kept to a minimum to ease the maintenance task. They can be categorized as follows:

- Data statement incompatibilities — Extensive use of labeled common is made, which allows presetting data items with data statements contained in a BLOCK DATA subroutine. There is a different BLOCK DATA subroutine for each machine.

- Word size is different for each machine. The following variables must be correctly set up in the BLOCK DATA subroutine so the assembler's character manipulating subroutines will work:

Name	1700 Value	6000 Value
BYTE1	$FF00_{16}$	77000000000000000000B
BYTE2	$00FF_{16}$	00777777777777777777B
CSHIFT	8	6
BLKPAD	0020_{16}	00555555555555555555B

- The CYBER 170/70/6000 is a faster machine than the CYBER 18. To help speed up the CYBER 18, all I/O routines in the CYBER 18 version make extensive use of the FORTRAN run-time monitor to make MSOS monitor calls to perform the actual I/O. The CYBER 170/70/6000 version uses FORTRAN I/O calls.

- Since mass storage addressing is different on the two machines, all mass storage I/O routines are unique for their respective machines.

- The CYBER 18 FORTRAN compiler and the Macro Assembler allow program identification material to be included as part of the PROGRAM, SUBROUTINE, FUNCTION, or NAM cards. This identification is then transferred to the relocatable binary decks that make up the assembler and is printed by the MSOS loader when the assembler is loaded. Each source deck in the assembler for the CYBER 18 version contains this identification, which will cause an error if the same deck is compiled on the CYBER 170/70/6000 FORTRAN-extended compiler.

- The PROGRAM card for the main routine of the assembler for the CYBER 170/70/6000 version defines all I/O files to be used during an assembly. This card will cause an error when read by the CYBER 18 FORTRAN compiler.

- All the assembler routines for the CYBER 170/70/6000 version must be compiled with the FTN (FORTRAN-extended) compiler. All assembler routines for the CYBER 18-17 version except CYBER 18-10/20/30 must be compiled with the standard FORTRAN compiler. The CYBER 18-10/20/30 must be assembled with the Macro Assembler.

The following programs are identical between the two versions of the assemblers except for the previously noted differences in the PROGRAM, SUBROUTINE, or FUNCTION cards.

Name	Function
ASMP17	Main routine
LIST	Format output listing
TABLE	Manipulate symbol table
PRINT1	Format assembled line, source, and diagnostics
PRINT2	Format comment line
PRINT3	Format first line of listing header
PRINT4	Format second line of listing header
PRINT5	Format number of errors in assembly
PRINT6	Format no-error message
PRINT7	Format blank lines
PRINT8	Format ORG and EQU listing output
PRINT9	Format copyright message
PRINT10	Format ZMAP and PMAP listing lines
SPLIT	Split source card into functional fields
PUTFLD	Put data field
GETFLD	Get data field
PUTCHR	Put character
GETCHR	Get character
BINHEX	Internal binary to external hexadecimal character conversion
BINASC	Internal binary to external decimal character conversion
VALUE	Find value of data item (either symbol or constant)
EVALU8	Evaluate address expression
NUMCON	Evaluate a constant
OPER8R	Check character for an operator (+ − * /)
IFIXIT	Convert double-precision value to integer
PATAPE	Format absolute object output
PRTAPE	Format relocatable object output
PMAP	Format origin map
ZMAP	Format zero map

Name	Function
CLEAR	Clear data buffer
PCARD	Format deadstart object output
CHKSUM	Calculate twos complement 16-bit checksum
A2SCMP	Perform twos complement, 16-bit arithmetic
DEDINS	Converts micro instructions to ASCII
PAKOUT	Formats and outputs lead start cards

The following programs are special for the particular machine on which they run:

Name	Function
BLOCK DATA	Contains data statements to preset labeled Common
LSTOUT	Writes to list output device
PONERD	Reads input for pass 1
DISKWT	Writes pass 1 output for subsequent input by pass 2
DISKRD	Reads pass 2 input
GETPAG	Gets a page of the micro-memory image
PBLANK	Punches blank leader on paper tape or writes EOF
PUNCH	Output routine for PRTAPE
APUNCH	Output routine for PATAPE
CDOUT	Output routine for PCARD
RSTP	Read symbol table page
ADD16	Perform 16-bit ones complement addition
COMP16	Perform 16-bit complement

The following routines run with the CYBER 170/70/6000 version of the assembler only:

Name	Function
BINCRD	Build CYBER 18-compatible formatted relocatable binary card image.
ADJUST	Convert characters from DISPLAY code to ASCII.
PACK	Pack 16-bit data words into 60-bit data words.

The following routines run with the CYBER 18 version of the assembler only:

Name	Function
SHIFT	Shift a word left or right
MP	Assembly language routine that puts the assembler itself into execution

INSTALLATION ON A CYBER 170/70/6000

The instructions necessary to install the Micro Assembler in the CYBER 170/70/6000 system are located in the installation handbooks for NOS and NOS/BE.

INSTALLATION ON A CYBER 18-17

The necessary assembler routines are compiled and the relocatable object code is ordered with control cards as shown in figure E-1. Figure E-1 is the load map of the assembler installation accomplished on an MSOS 4.1 system with the following logical unit assignments:

Device	LU	Contents
Magnetic tape	6	Assembler relocatable object decks with control cards
Mass memory	8	Scratch

The assembler is installed as a file on the CYBER 18 system to avoid loader overhead each time the assembler is executed. MP is an assembly language routine that is called by the control card *MP, which in turn reads in the assembler file and executes it.

Under MSOS 4, the following FORTRAN system routines must be available and must be loaded when the file is built:

FORTN

Q8PRMS

FXFL

FLOAT

PSSTOP

Q8PAND

Q8DBLE

Q8DFLT

DFLOTN

DUMVOL

DRSTOR

```
*LIBEDT
 LI
 IN
*K,I6,P8
 IN
*P,F
   MP      2339   7.5    04-19-74
   ASMP17  2936   15.00  00-24-74
   Q8Q8OS  33FA   16.0    09-22-74
   LIS1    33FA   15.00  00-24-74
   1ABLE   35FA   13.00  07-00-74
   RSTP    4099   16.1    09-22-74
   PRINT1  412A   15.00  00-24-74
   PRINT2  4197   13.00  07-00-74
   PRINT3  4203   13.00  07-00-74
   PRINT4  4242   9.0     04-16-74
   PRINT5  4240   13.00  07-00-74
   PRINT6  4268   9.0     04-16-74
   PRINT7  4277   9.0     04-16-74
   PRINT8  4292   13.00  07-00-74
   PRINT9  42F6   13.00  07-00-74
   PRNT10  4314   13.00  07-00-74
   LS10UT  4389   16.1    09-22-74
   SPLIT   44CA   13.00  07-00-74
   PUTFLD  4512
   GETFLD  453E
   PUTCHR  456A   9.0     04-16-74
   GETCHR  4583   9.0     04-16-74
   BINASC  45E3   9.0     04-16-74
   BINHEX  4654   9.0     04-16-74
   VALUE   4698   9.0     04-16-74
   EVALU8  46F7   13.00  07-00-74
   NUMCOM  47A0   9.0     04-16-74
   OPEROR  484E   9.0     04-16-74
   IFIXIT  4870   9.0     04-16-74
   POWERD  480A   13.00  07-00-74  2 READS
   OISKNT  498D   13.00  07-00-74
   OISKRD  49EA   13.00  07-00-74
   GETPAG  4A0F   13.00  07-00-74
   PATAPE  4A77   13.00  07-00-74
   PRTAPE  4A02   13.00  07-00-74
   PBLANK  4BC6   7.2     01-22-74  MP17
   PUNCH   4C38   7.1     01-22-74  MP17
   APUNCH  4C5F   7.1     01-22-74  MP17
   PMAP    4C83   13.00  07-00-74
   ZMAP    4CED   13.00  07-04-74
   CLEAR   4047
   PCARD   4067   13.00  07-04-74
   CODUT   4E18   13.00  07-04-74
   CHKSUM  4E65   13.00  07-00-74
   SHIFT   4EE1   7.6     01-25-74  MP17
   ADD16   4F18   7.2     01-22-74  MP17
   COMP16  4F2C   7.2     01-22-74  MP17
   A2SCMP  4F3F   16.0    09-22-74
   FORTN   4FA8   DECK-ID F01   3.2 FTN RUNTIME    SUMMARY-079
   Q8PRMS  5000   DECK-ID G01   3.2 FTN RUNTIME    SUMMARY-079
   FXFL    50F7   DECK-ID G06   3.2 FTN RUNTIME    SUMMARY-079
   FLOA1   515E   DECK-ID G13   3.2 FTN RUNTIME    SUMMARY-079
   PSSTOP  5380   DECK-ID H16   3.2 FTN RUNTIME    SUMMARY-079
   Q8PAND  53E7   DECK-ID H17   3.2 FTN RUNTIME    SUMMARY-085
   Q8DBLE  5449   DECK-ID K02   3.2 FTN RUNTIME    SUMMARY-079
   Q8DFLT  545F   DECK-ID K07   3.2 FTN RUNTIME    SUMMARY-079
   DFLOTN  5480   DECK-ID K12   3.2 FTN RUNTIME    SUMMARY-079
   DUMVOL  584A   DECK-ID K13   3.2 FTN RUNTIME    SUMMARY-079
   DRSTOR  587A   DECK-ID K14   3.2 FTN RUNTIME    SUMMARY-079
 IN
*K,I0
 IN
*N,ASMP,..8
 IN
*K,I6,P11
 IN
*L,MP
 IN
*Z
*CTO, MP ASSEMBLER IS NOW INSTALLED
*Z
```

Figure E-1. MSOS 4 Load Map

ASSEMBLER DEFAULT CODES F

Field	Conditions	Default	Code Decimal
F	(1) B not blank	B	15
	(2) A not blank, B blank	A	10
	(3) A and B both blank	Zero	12
A		X	3
B		X	3
D		NOP	0
S		NOP	0
C			0
M	C field is K= or N=	S	3
	C field is not K= or N=	S	1
T	Upper instruction and M field is S	*L	0
	Lower instruction and M field is S	U	1
	M field is R	U	1
	M field is J, C is constant	U	1
	M field is J, C is upper symbol	U	1
	M field is J, C is lower symbol	L	2

FORMAT OF MICRO-MEMORY IMAGE PAGES   G
ON MASS STORAGE

Word

1	Page number = 0, 1, 2, ..., 42
2	Binary zero

384 words — 382 words

191 32-bit instructions

16 bits of an instruction per word

Six 96-word sectors

384

385

192 words — 191 words

191 card numbers corresponding to the 191 micro instructions on this page

575

576	Binary zero
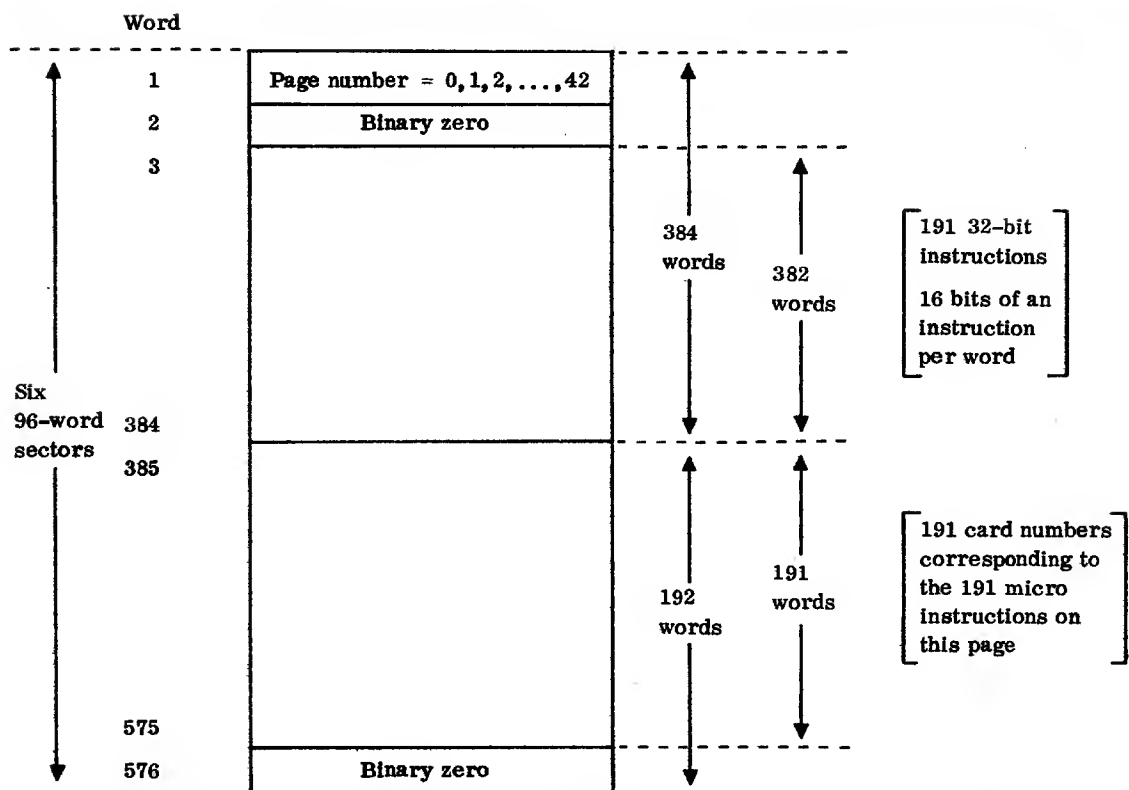
The following are equations to calculate page number, index to a micro instruction on the page, and index to the card number of the code that assembles into the micro instruction for a given micro instruction address:

$$PAGSIZ = 384$$

$$T = 0 \text{ if upper instruction of a word}$$

$$T = 1 \text{ if lower instruction of a word}$$

$$PMA = \text{micro instruction word address}$$

$$WORD = PMA*2+T \text{ (32-bit instruction number starting at zero)}$$

$$I = WORD*2+1 \text{ (16-bit half instruction number starting at one)}$$

J	=	PAGSIZ-2 (number of 16-bit half instructions per page)
RQPAGE	=	I/J (page number)
INDEX1	=	I-RQPAGE*J+2 (index to the first 16-bit half instruction)
INDEX2	=	INDEX1+1 (index to the second 16-bit half instruction)
CINDEX	=	INDEX1/2+PAGSIZ (index to the card number of the code that assembles into the micro instruction)

PAGE(CINDEX) =   0 ⇒ The corresponding micro instruction address is not used.

<   0 ⇒ The corresponding micro instruction address was the first instruction assembled following an ORG. The real card number is -PAGE(CINDEX).

>   0 ⇒ The instruction was assembled from card PAGE(CINDEX).

GLOSSARY

ALU	The portion of the computer which performs arithmetic and logical functions on two input quantities.
A/Q	A register, Q register or the combined A/Q register. The A and Q registers are shift registers.
A source	The first input to the ALU.
B source	The second input to the ALU.
Ones complement	The radix-minus-one complement in binary notation.
S1	An eight to one multiplexer used to select the A source.
S2	An eight to one multiplexer used to select the A source.
Twos complement	The radix complement in binary location.

ALLOCATION OF SCRATCH MASS MEMORY BY THE CYBER 18-17 VERSION OF THE ASSEMBLER   H

Top of
Scratch

Symbol Table Pages — 2500 words/page
    27 sectors/page
    500 symbols per page
    Page numbers start at 1
    20 pages maximum = 540 sectors maximum = 10,000 symbols
      maximum

Micro-memory image and corresponding card numbers
    576 words/page = 6 sectors/page
    191 32-bit instructions/page
    Maximum of 43 pages = Maximum of 258 sectors
    Page numbers start at 0
    Micro-memory image data is written by pass 2

Source card images written by pass 1, read by pass 2
    One sector per card

Scratch
Sector 1

INDEX

COMMENT SHEET

MANUAL TITLE   CONTROL DATA® CYBER Cross System Version 1 Micro Assembler

                 Reference Manual

PUBLICATION NO.   96836400   REVISION   C

FROM   NAME:

         BUSINESS
         ADDRESS:

COMMENTS: This form is not intended to be used as an order blank. Your evaluation of this manual will be
           welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or
           general comments may be made below. Please include page number to which your comment
           applies.

FOLD

- -

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
PUBLICATIONS AND GRAPHICS DIVISION
4455 EASTGATE MALL
LA JOLLA, CALIFORNIA   92037

- -

FOLD

CUT ALONG LINE

CONTROL DATA CORPORATION